فهرست

بخش اول

اصول پایه

به دنیای لینوکس و متن‌باز خوش آمدید. برای خواندن این کتاب داشتن یک پیش زمینه فنی لازم است. اما نیازی به داشتن دانش قبلی در مورد لینوکس و متن‌باز نیست. در قسمت اول مفاهیم پایه‌ای در مورد سیستم‌عامل متن‌باز، مدل توسعه و مراحل مجوزگیری آمده است. فصل اول دلایل نیاز به لینوکس و متن‌باز را از لحاظ تجاری بررسی می‌کند. فصل دوم کرنل لینوکس را به طور مفصل شرح می‌دهد. فصل سوم به تعریف متن‌باز و مجوزهای مربوطه می‌پردازد و در نهایت فصل چهارم دیدی کلی درمورد سرعت رشد و پیشرفت متن‌باز می‌دهد.

فصل اول

کسب و کار لینوکس و متن‌باز

لینوکس و متن‌باز فناوریها، ایده‌ها، مفاهیم ونمونه‌های بسیاری را شامل می‌شوند. لینوکس، سیستم-عامل جدید و عظیمی است که با استفاده از متدولوژی متن‌باز توسعه یافته است. برای درک لینوکس و فهم تأثیر توسعه و متن‌باز بر روی کسب و کار و سازمان، نگاهی گذرا به برخی مفاهیم، امری ضروری است. این فصل قصد دارد شما را با مفاهیم زیر آشنا کند:

- لینوکس کجا استفاده می‌شود و نحوه رشد آن چگونه است
- واژگان فنی که در فهم این کتاب مفید خواهد بود
- مزایای اصلی تجارت لینوکس و متن‌باز
- موانع اصلی محدود کننده رشد لینوکس
- نقش آفرینان اصلی در جامعه متن‌باز

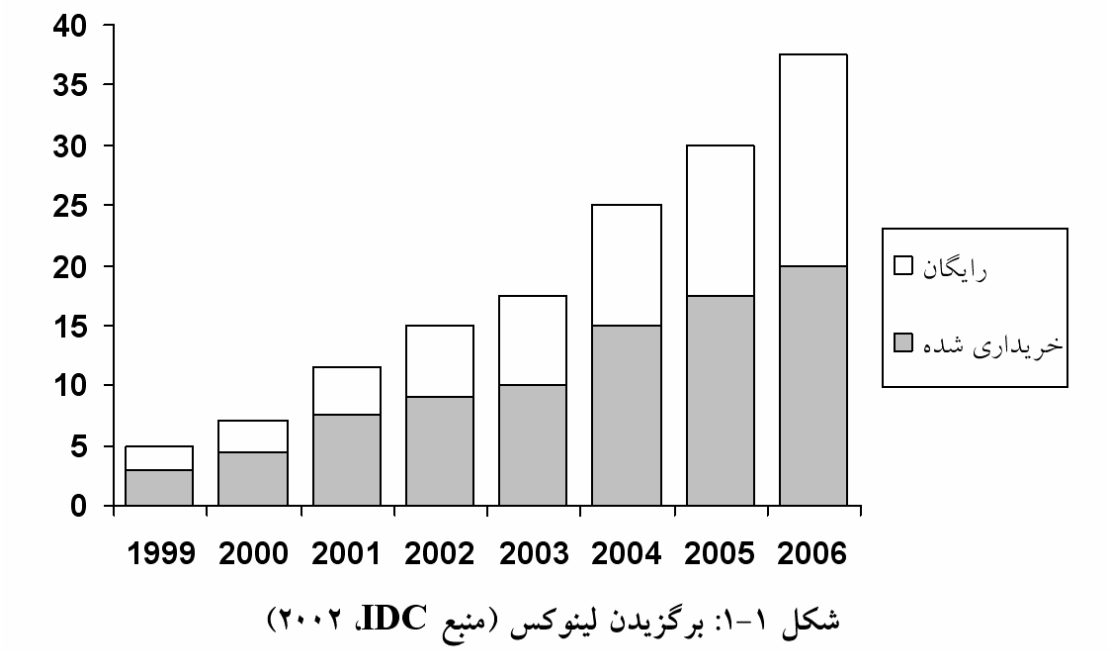
این فصل اهمیت لینوکس و متن‌باز را در تجارت بررسی می‌کند و مفاهیم پایه‌ای- که آشنایی با آنها برای خواندن این کتاب لازم است -را شرح می‌دهد.

پذیرش لینوکس

به طور معمول در بررسیهای مختلف، از نمودارهایی که نمایانگر بازده مورد نظر یک محصول یا خدمات در بازار مورد هدف هستند، استفاده می‌شود. این کار در مورد لینوکس نمی‌تواند خیلی دقیق باشد، چرا که محاسبه دقیق تعداد نسخه‌هایی از لینوکس که در حال استفاده هستند، غیرممکن است. در واقع این محاسبه برای لینوکس به واسطه متن‌باز بودن آن بسیار دشوار است.

درست است که امکان خرید نسخه‌های لینوکس وجود دارد، ولی شرکتها اغلب یک نسخه از لینوکس می‌خرند، آن را مطابق با نیازهای خود تغییر می‌دهند و سپس آن را به کار می‌گیرند. از طرف دیگر لینوکس را می‌توان از منابع مختلف به طور رایگان دریافت کرد. لینوکس در لوح فشرده همراه با برخی کتابها یا محصولات دیگر نیز عرضه می‌شود. همچنین برخی از کاربران برای به‌کارگیری چندین سیستم‌عامل مختلف روی یک کامپیوتر، سیستمهایشان را به صورت بوت دوگانه پیکربندی می‌کنند.

این موارد شمارش دقیق تعداد لینوکسهای مورد استفاده را دشوار می‌کند. در عین حال تحلیل‌گران زیادی مثل IDC نمودارهایی مانند نمودار شکل ۱-۱ را برای داشتن تقریب منتشر کرده‌اند. شکل ۱-۱ تعداد نسخه‌های لینوکس ـ هم رایگان و هم خریداری شده ـ را نشان می‌دهد.



شکل ۱-۱: برگزیدن لینوکس (منبع IDC، ۲۰۰۲)

همانطورکه اشاره شد، چون لینوکس را می‌توان رایگان بدست آورد، سود ناشی از این سیستم‌عامل عملاً بی معنی است. اما داشتن تقریبی از تعداد نسخه‌های آن، شاخص خوبی در زمینه موارد مرتبط با درآمد محصول، مانند کارخواه‌ها، نرم‌افزارها و موارد دیگر خواهد بود. نرخ سرعت رشد لینوکس و متوقف نشدن رشد آن در آینده نزدیک نیز از روی نمودار مشخص است. توجه کنید که نسخه‌های رایگان و خریداری شده هر دو باید لحاظ شوند. به‌طور منطقی، از این نمودار می‌توان این‌گونه

برداشت کرد که شرکتهای مختلف، سرمایه‌گذاری محصولات آینده خود را در بخش فنی را به سمت لینوکس پیش می‌برند تا از مزایای این رشد بهره‌مند شوند.

نگاهی گذرا به لینوکس و واژگان متن‌باز

وقتی شما خواندن این کتاب را به اتمام برسانید، درک بهتری از لینوکس، متن‌باز و نحوه به کارگیری آنها در کسب و کار خود پیدا خواهید کرد. در هر صورت فهم سطحی یک سری مفاهیم، به درک اولیه شما کمک خواهد کرد و همان طور که با کتاب پیش می‌رویم، این مفاهیم با جزئیات بیشتر بیان خواهند شد:

- **کرنل**: کرنل، بخش اصلی سیستم‌عامل را تشکیل می‌دهد که کار آن مدیریت منابع اصلی (مثل حافظه، پروسه‌ها و سایر موارد) سیستم است.
- **گنو (GNU)**: این لغت از کنار هم قراردادن حروف اول GNU'S NOT UNIX (گنو یونیکس نیست) شکل گرفته است. چون لینوکس به خودی خود کرنل سیستم‌عامل است، سیستم گنو بخشهای دیگری را (مانند کامپایلرها، ویرایشگرها و سایر موارد) که مورد نیاز یک سیستم کامل هستند، ارائه می‌دهد و لینوکس با استفاده از این ابزارها تبدیل به یک سیستم‌عامل می‌شود. به همین دلیل است که لینوکس را معمولاً گنو/ لینوکس می‌نامند و این مسأله مهم همواره باید در نظر گرفته شود.
- **توزیع (نسخه خاص)**: شرکتهای مختلفی که برنامه‌های کاربردی برای لینوکس می‌نویسند، اغلب با استفاده از توزیعهای رایج لینوکس این کار را انجام می‌دهند. برخی از رایج‌ترین نسخه‌های لینوکس عبارتند از: SuSE، Redhat، Debian، Red Flag Connectiva، Mandrake. عرضه‌کنندگان توزیعها، کرنل لینوکس را با بخشهای مختلفی از سیستم گنو و برنامه‌های متن‌باز دیگر ترکیب می‌کنند و پس از اعمال اصلاحات مورد نظر خود، مجموعهٔ حاصل را به صورت یک سیستم‌عامل کامل و تست شده عرضه می‌کنند. این توزیعهای

مختلف همگی لینوکس هستند، ولی هر کدام برای یک یا چند امر خاص طراحی شده‌اند و قابلیت خاص خودشان را به کاربران ارائه می‌دهند.

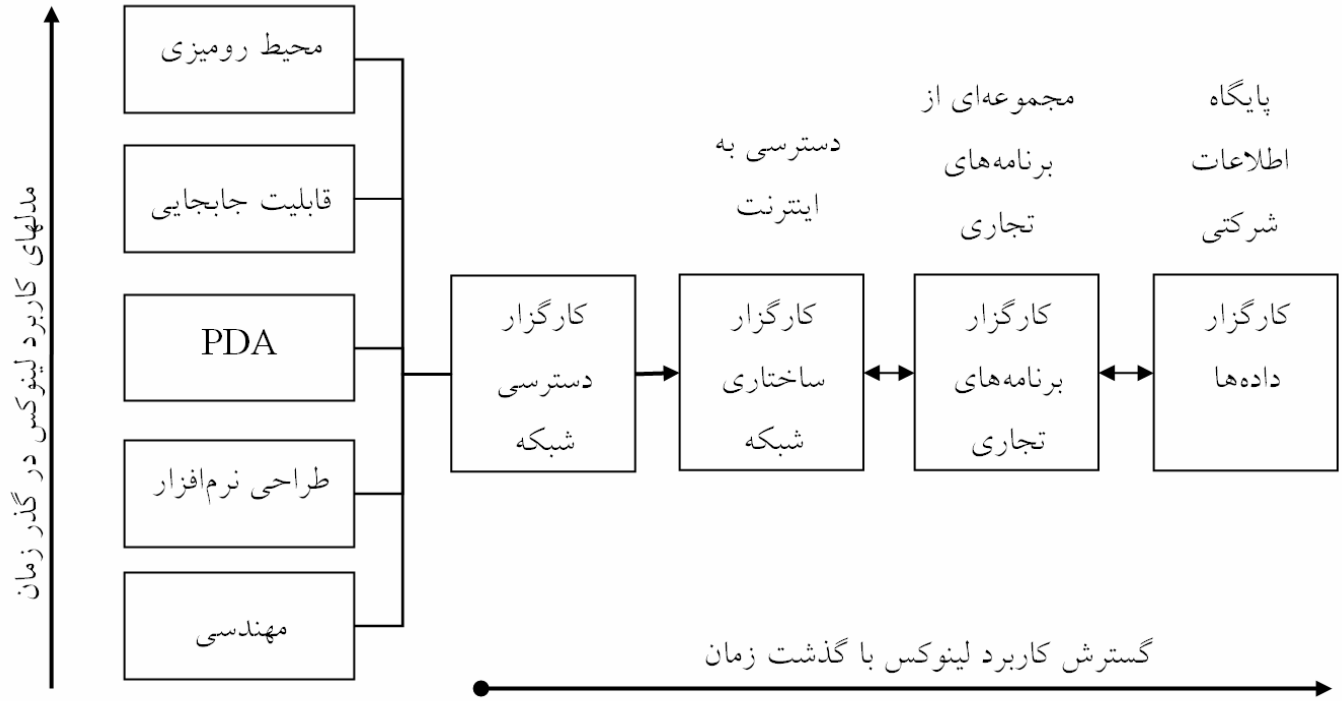
- **بستهٔ نرم‌افزاری**: هرکدام از برنامه‌هایی که روی سیستم‌عامل لینوکس نصب می‌شوند، بصورت بسته‌های نرم‌افزاری نیز توزیع می‌شوند. عموماً یک بسته نرم‌افزاری به مجموعه‌ای از فایلها که برای انجام یک کار خاص در کنار هم قرار می‌گیرند، اطلاق می‌شود. چون بسته‌های نرم‌افزاری مختلف اغلب از یک سری فایل بطور مشترک استفاده می‌کنند، این وابستگیها با کمک ابزارهای خاص نصب کنترل و مدیریت می‌شوند.
- **نرم‌افزارهای آزاد**: مفهوم نرم‌افزار آزاد که با تاسیس "بنیاد نرم‌افزار آزاد" (FSF) شکل گرفت به این معناست که نرم‌افزارها باید در اختیار دیگران قرار داده شوند و هرکس باید دسترسی یکسانی به کد منبع یا طراحی سیستم داشته باشد.
- **متن‌باز**: این واژه عموماً در دو زمینه به کار می‌رود: یکی به عنوان یک اصطلاح بازاریابی برای نرم‌افزار آزاد و مورد دیگر به عنوان یک نوع متدولوژی توسعه که جزئیات آن در بخش سوم بررسی خواهد شد. از آنجایی که پیشگامان حرکت نرم‌افزارهای آزاد مخالف استفاده از اصطلاح آزاد هستند، این واژه برای از بین بردن تصور غلط نرم‌افزار رایگان به جای نرم‌افزار آزاد ساخته شد. متن‌باز یک مجوز نیست، بلکه مجموعه شرایطی را تعیین می‌کند که مجوزها برای متن‌باز بودن، باید لحاظ کنند.
- **انجمن**: به هر مجموعه‌ای از توسعه‌دهندگان نرم‌افزار (برنامه‌نویسان) که با همکاری یکدیگر بر روی یک پروژه نرم‌افزاری کار می‌کنند، انجمن گفته می‌شود. این افراد می‌توانند دانش‌آموزان، افراد با علائق خاص، برنامه‌نویسان در استخدام شرکتها، رقیبان و یا مشتریان باشند.

- **نگهدارنده**: یک فرد، کمیته و یا مجموعه‌ای از مدیران که در مورد قبول یا رد تغییرات در کد پروژه متن‌باز تصمیم می‌گیرند. نگهدارنده را می‌توان مدیر پروژه تلقی کرد. معروف‌ترین نگهدارنده در دنیای متن‌باز، لینوس توروالدز - طراح کرنل لینوکس - می‌باشد.
- **GPL**: این واژه مخفف مجوز کلی و عمومی گنو می‌باشد. GPL به طور مفصل در فصل سوم شرح داده شده است. تا رسیدن به آن فصل دانستن این که GPL، پرکاربردترین مجوز متن‌باز است، کافی است. GPL مجوزی است که کرنل لینوکس را تحت پوشش خود دارد و مقرر می‌کند که هر تغییر در کد منبع باید باید به انجمن برگردانده شود.

فهم این واژگان برای شروع کار کافی است. همان طور که کتاب را می‌خوانید به عقب برگردید و مطمئن شوید که واژه‌ها را خوب فهمیده‌اید.

حجم کاری لینوکس

شکل ۱-۲ موارد کاربرد لینوکس و نحوه گسترش آن را در گذر زمان نشان می‌دهد. لینوکس در سمت کارگزار قویترین ابزار در "لبه شبکه" (نقطه دسترسی به شبکه) است که این، عمدتاً به واسطه ترکیب پرقدرت سیستم‌عامل لینوکس و سرور وب آپاچی محقق شده است. کاربردهای لینوکس همراه با رشد آن به تدریج به سمت کاربردهای زیربنایی گسترش می‌یابد.



شکل ۱-۲: رشد حجم کار لینوکس با گذشت زمان

بحثی که در اینجا پیش می‌آید این است که این تحول چقدر طول می‌کشد. در سمت کارخواه، لینوکس با داشتن سابقه زیاد به عنوان یک سیستم‌عامل شبه یونیکس، طبیعتاً به واسطه ایستگاه‌های کاری مهندسی و سیستمهای توسعه برنامه‌های کاربردی شهرت پیدا کرده است. لینوکس همچنین در عرصهٔ وسایل توکار[1] (embedded) نفوذ کرده است.

سؤالی که مطرح می‌شود این است که آیا لینوکس جای خود را روی میزهای کار اداری به عنوان یک سیستم‌عامل پایدار باز خواهد کرد یا خیر؟

مزایای تجاری

لینوکس به سرعت از یک وسیله سرگرمی به یک محیط مطمئن و قابل استفاده در بسیاری از کسب و کارها تغییر نقش داده است. یکی از مهمترین دلایل گسترش آن در دنیای تجارت، توسعه چشمگیر اینترنت بود. کارگزارهای اینترنت، لینوکس را راه‌حل مناسبی برای به‌کارگیری سیستمهای

[1] embedded

عمومی شده یافتند. مجوز سیستم‌عامل یا برنامه‌های اصلی کاربردی متن‌باز مانند وب سرور آپاچی و یا برنامه‌های دیگر مورد نیاز اینترنت که تحت لینوکس اجرا می‌شوند، به گونه‌ای است که هزینه‌ای برای آنها دربرنداشت. هم اکنون محیطهای تجاری لینوکس را به عنوان یک جایگزین قابل اطمینان نسبت به محیطهای دیگر از لحاظ هزینه، منابع و رویکردهای کنترلی می‌دانند. در ادامه برخی از مزایای اقتصادی لینوکس آورده شده است.

هزینه

این واقعیت که لینوکس می‌تواند بدون پرداخت حق‌الامتیاز بطور رایگان عرضه شود، یکی از مزایای تجاری لینوکس به شمار می‌رود. به هر حال باید توجه داشت که اگر چه توسعه و به‌کارگیری لینوکس بدون پرداخت هزینه مجوز ممکن است، این بدان معنی نیست که لینوکس و یا هر سیستم‌عامل دیگر از لحاظ هزینه‌های مربوط به نصب، نگهداری، پشتیبانی و آموزش نیز رایگان می‌باشد. در فصل ششم، ما نگاه مفصلی به هزینه‌های جانبی لینوکس و نرم‌افزارهای متن‌باز دیگر خواهیم داشت.

در دسترس بودن مطالب آموزشی

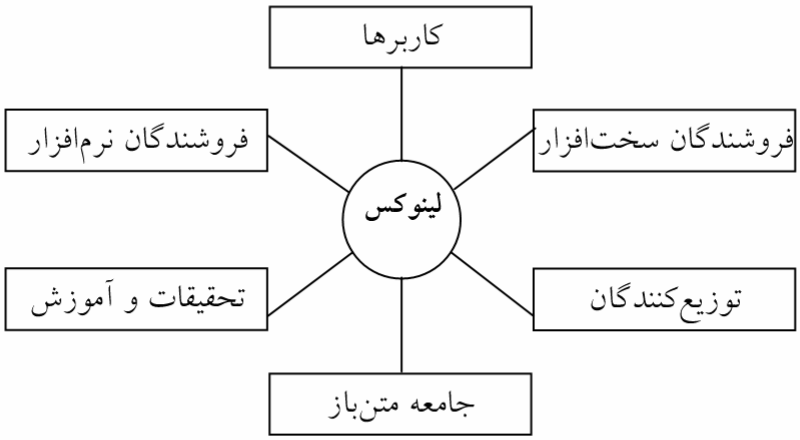
لینوکس بیش از ده سال قدمت دارد، بنابراین منابع آموزشی بسیاری در این زمینه موجود است. هزینه‌های کم لینوکس، بسیاری از مؤسسه‌های تحقیق و پژوهش مثل دانشگاه واترلو در آنتاریوی کانادا، دانشگاه ایلینویز در اوربانا شامپین، NCSA[1] و بسیاری از مؤسسه‌های دیگر را به فکر استفاده از لینوکس انداخته است. این رویدادها در طول چند سال رخ داده است. مثلاً گروه ESIEE که یکی از مراکز آموزش علمی و فنی پیشرفته فرانسه است، به مدت سه سال است که از لینوکس استفاده می‌کند. زیرا لینوکس متن‌باز است. انگیزه آنها از این کار همانگونه که خودشان اظهار داشته‌اند، بسیار ساده است: گروه آنها برای جامعه توسعه می‌یابد و جامعه برای گروه آنها توسعه می‌یابد. این

[1] مرکز ملی کاربردهای ابررایانه‌ای

کار یک نوع برگشت سرمایه برای آنهاست. دانشگاه[1] در استرالیا، تمام لابراتوارهای یونیکس خود را ظرف مدت یک سال به لینوکس تبدیل کرد و در نتیجه فارغ‌التحصیلان جدید به طور کامل تحت سیستم‌عامل لینوکس آموزش دیدند.

نسل جدید برنامه‌نویسان نیز درحال کسب تجربه تحت لینوکس و واسطهای برنامه‌نویسی کاربردی (API) آن هستند. این نسل جدید با برنامه‌های کاربردی متن‌باز مشهوری مثل وب سرور آپاچی نیز آشنا است.

شکل ۱-۳ گروه‌های اصلی دخیل در جنبش لینوکس را نشان می‌دهد. با رشد و توسعه سازمان و نیاز به داشتن نیروی کار مستعد برای راهبری زیر ساخت شبکه، کارگزارهای برنامه‌های کاربردی و مرکز داده‌ها، و افراد مستعد در زمینه لینوکس استعدادهای لینوکس یکی از دردسترس‌ترین منابع مورد نیاز شما خواهد بود.



شکل ۱-۳: افراد سهیم در لینوکس

پشتیبانی

یکی از بحثهای داغ در مورد لینوکس این است که پشتیبانی از طریق چه مرجعی صورت گیرد. یک طرف این بحث بر آن است که چون لینوکس توسط جمعی از توسعه‌دهندگان -که ارتباط کمی با

[1] New south wales

هم دارند - نگهداری می‌شود، ضمانت و پشتیبانی آن مورد شک است. طرف دیگر عقیده دارند چون کد منبع در دسترس همه است، هر کسی می‌تواند به نوعی پشتیبانی را فراهم کند و این پشتیبانی، هزینه کمتر و اطمینان بیشتری به همراه دارد. اینکه شما کدام طرف بحث را بپذیرید بستگی به نوع سازمان فناوری اطلاعات و تجربه قبلی شما در تعامل با عرضه‌کنندگان مختلف خدمات پشتیبانی دارد. به هر حال شما در انتخاب نوع پشتیبانی، قدرت انتخاب دارید. فصل هشتم به جزئیات راه حلهای پیش رو برای پشتیبانی و مزایای هریک خواهد پرداخت.

توسعه نرم‌افزار

لینوکس برای تیم برنامه‌نویسی Fluent اصلی‌ترین سکو محسوب می‌شود. Fluent یکی از شرکتهای پیشرو در ارائه نرم‌افزارهای مهندسی بکمک کامپیوتر[1] است. لینوکس، سکوی پایدار و قدرتمندی با گنجینه‌ای از ابزارهای رایگان را در اختیار برنامه‌نویسان این تیم قرار داده است. علاوه بر این، برنامه‌نویسان از طریق شبیه‌ساز VMware که تحت لینوکس اجرا می‌شود، می‌توانند به برنامه‌های کاربردی ویندوز دسترسی داشته باشند. VMware در واقع برنامه‌ای است (در نسخه‌های تحت ویندوز و لینوکس) که یک پردازنده اینتل را شبیه‌سازی می‌کند و به برنامه‌نویسان امکان دسترسی به سیستم‌عاملهای دیگر مثل، ویندوز و یونیکس را در یک ماشین می‌دهد. آمار نشان می‌دهد که ویندوز با در اختیار داشتن ویژوال استودیو، گوی سبقت را در بازار توسعه نرم‌افزار ربوده است. با این حال اگر بازار لینوکس را تقسیم‌بندی کنیم، می‌بینیم که توسعه نرم‌افزار در حال حرکت به سمت لینوکس است که یکی از دلایل آن بازار آموزش و تحقیقات می‌باشد که قبلاً به آن اشاره شد.

ارتقاء

هر ساله شرکتها، برای داشتن خروجی بهتر میلیونها دلار صرف ارتقاء برنامه‌های کاربردی یا محیطهای برنامه‌نویسی می‌کنند. در بسیاری از این موارد انگیزه ارتقاء به خاطر نیاز به مجوز و سخت‌افزاری است که از طرف عرضه‌کنندگان اعمال می‌شود. با لینوکس همه چیز کنترل شده

[1] Camputer aided engineering

است. شما می‌توانید هزینه پشتیبانی و ارتقاء را محاسبه کرده و سپس در مورد اعمال تغییرات تصمیم بگیرید. در حالت کلی کاربران تن به ارتقاء سیستم نمی‌دهند، مگر اینکه در نهایت منجر به سود تجاری بیشتر شود و سرمایه صرف شده برای ارتقاء و خطرات آن درآمدی هم داشته باشد. یکی از زیباییهای متن‌باز این است که تصمیم‌گیرنده شما هستید، نه عرضه‌کننده. سخت‌افزار نیز به تبع ارتقاء نرم‌افزار، ارتقاء خواهد یافت. در جامعه متن‌باز، برنامه‌نویسان زیادی از سیستمهایی استفاده کرده‌اند که از عمر آنها بیش از ۱۰ سال می‌گذرد و از کار افتاده محسوب می‌شوند.

شرکتها به خوبی می‌دانند که هر تغییری، ریسکها و فرصتهایی دربردارد که مانند هر فعالیت دیگری در سازمان، نیاز به اندازه‌گیری و کیفیت‌سنجی دارد. اگر شما قصد به‌کارگیری محیط کاری لینوکس را داشته باشید، هیچ مجوزی در دنیای لینوکس و متن‌باز شما را مجبور به تغییر، افول یا ارتقاء آن محیط نخواهد کرد. شما کنترل کاملی بر محیط خود دارید. بنابراین موفقیت یا شکست شبکه کاری و در نهایت کسب و کار خود را خودتان کنترل می‌کنید.

موانع رشد لینوکس

درست است که لینوکس یک رویکرد مناسب برای صرف هزینه کمتر، بالا بردن کارایی و شریک شدن با جامعه بزرگی از برنامه‌نویسان می‌باشد، اما لینوکس اکسیری نیست که پاسخگوی همه کاستیهای فناوری اطلاعات باشد. در زیر برخی از موانعی که بر سر راه لینوکس قرار دارد، آورده شده است. در عین حال تلاشهای بسیاری برای از بین بردن موانع رشد لینوکس در دنیای فناوریهای پیشرفته می‌شود. زمان، بسیاری از این موانع را از سر راه برخواهد داشت.

دردسترس بودن برنامه های کاربردی

لینوکس بسیاری از موفقیتهای خود را مرهون وب سرور آپاچی است. طبق تحقیقات صورت گرفته توسط Netcraft در اواخر سال ۲۰۰۱ وب سرور آپاچی توسط بیش از پنجاه درصد از سایتهای شبکه استفاده می‌شده است. ترکیب لینوکس و آپاچی، یک محیط مقرون به صرفه را با کارایی بالا- که به سادگی قابل گسترش است- برای کارگزارها و مؤسسات به وجود می‌آورد.

علاوه بر آپاچی، برنامه‌های کاربردی زیادی توسط انجمنهای آموزشی و تحقیقاتی و یا شرکتهای مختلف توسعه پیدا کرده‌اند.

مسأله کلاسیک مرغ و تخم‌مرغ برای محیطهای زیادی صادق است و لینوکس از این قاعده مستثنی نیست. درحالی‌که شرکتهای زیادی منتظر برنامه‌های کاربردی قبل از توسعه لینوکس هستند، تولیدکنندگان نرم‌افزار منتظر مشتریان هستند تا لینوکس را بکارگیرند، سپس برنامه‌های آن را تولید کنند.

ریسک تجاری

لینوکس و اغلب نرم‌افزارهای متن‌باز ریسکهای تجاری خاص خود را به همراه دارند. پیاده‌سازی هر فناوری جدیدی در سازمان با خطرهایی همراه است که باید مدیریت شوند. لینوکس و متن‌باز نیز از این قاعده مستثنی نیستند. فرآیندهای تجاری که از قبل موجود نیستند باید شکل گیرند. شرکتهای بسیاری درک کرده‌اند که لینوکس از عهده مدیریت این ریسکها و افزودن بر سرمایه اقتصادی به خوبی برمی‌آید. قسمت سوم این کتاب به بررسی این ریسکها و نحوهٔ مدیریت آنها می‌پردازد. در واقع متن‌باز فرصتهای جدیدی برای افزایش کارایی و تولید محصول و جلب رضایت مشتری به وجود می‌آورد.

نقش آفرینان متن‌باز

جامعه متن‌باز متشکل از افرادی است که در کل جهان روی پروژه‌های متن‌باز کار می‌کنند. بعضی از این افراد این کار را به عنوان یک وظیفه و برخی نیز صرفاً از روی علاقه انجام می‌دهند. بعضی به علت اعتقاد به متن‌باز و بعضی برای پیدا کردن جواب مسایل خود به این کار می‌پردازند. تعدادی از این برنامه‌نویسان عضو انجمنهای آموزشی و تحقیقاتی هستند. جدول ۱-۴ نمودار زمانی رویدادهای مهم و اصلی را-که از زمان حیات لینوکس و متن‌باز رخ داده است - نشان می‌دهد.

در قسمت سوم این کتاب ما دلایل تشویق کارمندان توسط مدیران در روی آوردن به پروژه‌های متن‌باز را شرح خواهیم داد. تاکنون افراد برجسته و رویدادهای بارزی که ارزش شناخته‌شدن داشته

باشند و فهمیدن تأثیر آنها در جامعه متن‌باز مفید باشد، خیلی زیاد نبوده است. در عین حال آوردن نام همهٔ کسانی که به نحوی در این کار نقش داشته‌اند نیز ممکن نیست. اما باید این نکته را در نظر داشت، افرادی هستند که نقش تعیین‌کننده‌ای در این کار داشته‌اند، ولی نام آنها ذکر نشده است. به تدریج متوجه خواهید شد که متن‌باز در واقع یک بازی تأثیرگذاری و ایجاد روابط است. به محض اینکه شما جوامع مفید برای کسب و کارتان را شناختید، به ارزش صرف وقت و انرژی برای برقراری ارتباط با رهبران و اعضاء آن جوامع پی خواهید برد.

۱۹۹۴	بنیاد نرم‌افزار آزاد تشکیل شد
۱۹۸۹	FSF اولین نسخه GPL را عرضه کرد
ژوئن ۱۹۹۱	FSF دومین نسخه GPL را عرضه کرد
سپتامبر ۱۹۹۱	اولین کد لینوکس عرضه شد
مارس ۱۹۹۴	لینوکس ۱،۰ عرضه شد
۱۹۹۴	شرکت RedHat تأسیس شد
۱۹۹۵	گروهی از برنامه‌نویسان دور هم جمع شدند تا برروی سرور وب آپاچی کار کنند
ژوئن ۱۹۹۶	لینوکس ۲،۰ عرضه شد
اکتبر ۱۹۹۶	پروژه میزکار KDE آغاز شد
مه ۱۹۹۷	مقاله "کاتدرال و بازار" منتشر شد
۱۹۹۷	پروژه میزکار GNOME آغاز شد
فوریه ۱۹۹۸	واژه متن‌باز رسمی شد
ژانویه ۲۰۰۱	نسخه ۲،۴ لینوکس عرضه شد

شکل ۱-۴ نمودار زمانی وقایع لینوکس و متن‌باز

Linus Torvalds (لینوس تورووالدز): وی احتمالاً مشهورترین شخصیت در جهان لینوکس و متن‌باز است. تورووالدز طراح و نگهدارنده کرنل لینوکس است. او اجزای مستقل از سکوی (سخت-افزار) کرنل را نگهداری می‌کند و اجزای هسته مرکزی کرنل را -که مختص معماری اینتل هستند،- نگهداری می‌کند. واسطهای لینوکس برای معماریهای دیگر نیز به وسیله اشخاص یا گروه‌های جداگانه نگهداری می‌شود.

Alan Cox: آلن تا پاییز سال ۲۰۰۱ جانشین اول لینوس محسوب می‌شد. در فصل دوم خواهید دید که لینوس، کرنل لینوکس را -که هم اکنون در حال توسعه است- نگهداری می‌کرد. درحالیکه آلن کرنل عرضه شده را محافظت می‌نمود. طبق یک قاعده جامعه متن‌باز، هر گاه آلن اراده می‌کرد، می‌توانست روی موضوع دیگری کار کند. او ابتدا یک جانشین برای خود انتخاب کرد. اگر چه مدت زیادی نیست که آلن به این کار مشغول است، نفوذ قابل توجهی در جامعه دارد و از اعتبار و احترام فراوانی برخوردار است.

Marcelo Tosatti: لینوز و آلن وظیفه نگهداری کرنل پایدار لینوکس (کرنل درحال تولید) را به مارچلو سپردند. وی یکی از توسعه‌دهندگان کرنل لینوکس است که لیاقت بالایی در کار با جامعه برنامه‌نویسان از خود نشان داده است.

Dave Miller: میلر مسئول اعمال تغییرات صورت گرفته در پشته‌های شبکه در کرنل لینوکس است و در مورد یا رد این تغییرات تصمیم می‌گیرد. میلر را می‌توان نگهدارنده بخش شبکه کرنل لینوکس دانست.

Richard Stallman: مؤسس FSF و طراح مجوز GNU GPL می‌باشد. ریچارد همچنین نقش تعیین‌کننده‌ای در جنبش نرم‌افزارهای آزاد دارد. افراد زیادی از نظرات و راهنماییهای او در قسمتهای مختلف GPL بهره می‌گیرند.

Bruce Perens: مدیر پروژه Debian -که در فصلهای بعدی به تفصیل شرح داده خواهد شد - و تدوین‌کننده یک قرارداد اجتماعی برای جا انداختن مجموعه اصول در کار با نرم‌افزارهای آزاد می‌باشد. وقتی واژه "متن‌باز" شکل گرفت، این قرارداد، "تعریف متن‌باز" (OSD) نام گرفت. به همین دلیل بروس را به عنوان مؤلف اصلی آن می‌شناسند. فصل سوم در این باره توضیحات بیشتری می‌دهد.

Eric Raymond: او به عنوان انسان‌شناس و رهبر فکری جامعه متن‌باز شناخته شده است. شهرت اریک به خاطر مقاله‌اش با عنوان "کاتدرال[1] و بازار" است. این مقاله فرایند توسعه متن‌باز، علت پیشرفت آن و نحوه پایداری و استمرار آن را شرح می‌دهد. بخش سوم این کتاب درباره این موارد بحث می‌کند.

Larry Augustin : لَری مدیر اجرایی شرکت نرم‌افزاری VA است. شرکت نرم‌افزاری VA در ابتدا با نام سیستمهای لینوکس VA شکل گرفت. لری از اولین افرادی بود که به اهمیت لینوکس و متن‌باز پی برد. او اولین شرکت سخت‌افزاری را که از لینوکس پشتیبانی می‌کرد ساخت. زمانی که شرکتهایی مثل HP، IBM و Dell سخت‌افزارهایی برای پشتیبانی لینوکس عرضه می‌کردند، لری مدل تجاری شرکت خود را به فروش نرم‌افزارهایی که فرآیند توسعه متن‌باز را در شرکتها پشتیبانی می‌کردند، تغییر داد.

Tim O'reilly: او به عنوان رئیس انتشارات O'reilly جایگاه خاصی در جنبش متن‌باز دارد. شبکه کتابهای O'reilly فناوریهای اصلی متن‌باز مثل Perl و موارد دیگری را که کاربرد وسیعی در اینترنت دارند، پوشش می‌دهد. او با چاپ افکار و ایده‌های جامعه متن‌باز، نقش سخنگوی این جامعه را ایفا می‌کند.

Brian Behlendorf: برایان یکی از بنیانگذاران مؤسسهٔ نرم‌افزاری آپاچی است. از آنجایی که آپاچی و متن‌باز در ارتباط با هم هستند، او نه تنها تأثیر به سزایی بر رشد آپاچی داشته است، بلکه در جامعه متن‌باز نیز به شدت تأثیرگذار بوده است.

Michael Tiemann: مایکل اولین شرکت بازرگانی مبتنی بر مدل تجاری متن‌باز را با عنوان Cygnus تأسیس کرد. Cygnus زیرساخت لازم برای یکی از اجزای کلیدی لینوکس، یعنی کامپایلرها را ارائه کرد. وی همچنین در تعریف مدلهای تجاری مبتنی بر تأمین پشتیبانی و ارائه خدمات برای نرم‌افزارهای آزاد، پیشرو بود. Cygnus در سال ۲۰۰۰ به RedHat پیوست.

---

[1] به کلیساهای مرکزی کاتدرال گویند که اغلب از ساختمانی بزرگ با معماری منحصر بفرد تشکیل شده‌اند.

Bob Young: اعتباری که باب امروزه نموده، ناشی از موفقیت محبوب‌ترین توزیع لینوکس یعنی Red Hat می‌باشد. باب برای ایجاد یک محیط تجاری قوی با استفاده از سیستم‌عامل تجاری لینوکس، از استعداد بازاریابی خود استفاده کرد. امروزه او یکی از رهبران اصلی شرکت RedHat است و در اغلب رویدادهای لینوکس و متن‌باز سخنرانی می‌کند.

Jeremy Allison: جِرِمی یکی از مدیران پروژه سامبا[1] (SAMBA) است. همان گونه که خواهید دید، سامبا یکی از نرم‌افزار کلیدی برای برقراری ارتباط بین جهان ویندوز و یونیکس/لینوکس است. سامبا یکی از فناوریهایی است که به همراه اکثر توزیعهای لینوکس عرضه می‌شود.

Miguel de Icaza: میگل مؤسس و مدیر ارشد فناوری شرکت Ximian است. او سالها مدیریت پروژه توسعهٔ محیط رومیزی گنو را بر عخده داشت و اکنون پروژه متن‌باز دیگری با نام Mono را رهبری می‌کند. این پروژه، پیاده‌سازی زیرمجموعه‌ای از محیط کاری دات نت (Net.) مایکروسافت، به صورت متن‌باز است.[2]

جمع‌بندی

اکنون با خواندن این مقدمه، شما اطلاعات کافی را برای مطالعه قسمتهای بعدی کتاب دارید. در ادامه خواهید دید که لینوکس فرصتهای بسیاری برای کاهش هزینه‌های فناوری اطلاعات در اختیار شما می‌گذارد. در عین حال نباید مدیریت خطرهای تجاری ناشی از آن را نادیده گرفت. با فهم واژگان اصلی و شناخت نقش‌آفرینان این حوزه، می‌توان به بررسی نحوه بهره‌مندی از مزایای متن‌باز و مدیریت خطر پرداخت.

فصلهای بعدی بخش اول، دانش شما را در مورد سیستم‌عامل لینوکس گسترش می‌دهند و جزئیات بیشتری از متن‌باز و مجوز مربوطه را روشن می‌کنند. این بخش با ارائه یک دید کلی نسبت به

---

[1] SAMBA

[2] Ximian در سال ۲۰۰۴ به شرکت Novell ملحق شد و Mono نیز تحت نظارت این شرکت هدایت می‌شود.

جوامع متن‌بازی که احتمال مواجهه و کار با آنها بیشتر است، به پایان می‌رسد. در بخش دوم نگاهی به جنبه عملی لینوکس می‌اندازیم. این بخش شما را تشویق به بررسی نیازهای شرکت می‌کند و اطلاعاتی درمورد پشتیبانی، نگهداری و هزینه‌های لینوکس در اختیارتان قرار می‌دهد. بخش سوم به بررسی مدیریت خطرها و فرصتهای متن‌باز می‌پردازد. این بخش علاوه بر شرح نحوه کار با پروژه‌های متن‌باز، نحوه پیاده‌سازی متدلوژی توسعه آن در محیط کار را نیز شرح می‌دهد.

فصل دوم

لینوکس - قلب سیستم‌عامل

معمولاً منظور از "لینوکس" کل سیستم‌عامل است و در اکثر موارد چنین تلقی از آن اشکالی ندارد. هر چند که لینوکس در واقع بخش کوچکی از یک سیستم‌عامل کاری را تشکیل می‌دهد. از نظر فنی، لینوکس فقط کرنل است؛ بخشی که باعث می‌شود سیستم به درستی به وظایف خود را عمل کند. در کنار کرنل اجزای دیگری وجود دارند که یک سیستم‌عامل را کامل می‌کنند. ابزارهایی که به عنوان بخشی از سیستم‌عامل لینوکس ارائه می‌شوند و در کنار هم را تبدیل به یک سیستم‌عامل کامل می‌کنند، سیستم گنو نامیده می‌شوند.

از این رو است که ریچارد استالمن ادعا می‌کند نام لینوکس برای سیستم‌عامل اشتباه است، بلکه باید کل سیستم را گنو/لینوکس نامید. مؤلفه‌های دیگری مثل سیستم پنجره‌ای X11، واسطهای میزکار، ابزارهای به اشتراک گذاردن فایلها و سایر ابزارها نیز باید به لینوکس اضافه شوند که با این حساب نام گنو/لینوکس هم کاملاً دقیق نیست. از نظر قانونی، لینوکس علامت تجاری ثبت شده برای "نرم‌افزار سیستم‌عامل کامپیوتر برای تسهیل کاربرد کامپیوتر" می‌باشد. لینوس توروالدز مالک این علامت تجاری است.

تصمیم به توسعه لینوکس در سازمان، نیازمند درک درست مؤلفه‌های لینوکس است. بنابراین اهداف این فصل درک مطالب زیر می‌باشد:

- یک دید ساده از سیستم‌عامل
- کرنل لینوکس و تفاوتهای آن با انواع دیگر کرنلها
- چگونگی توسعه و به‌روزرسانی کرنل لینوکس و سیستم نسخه‌بندی آن
- نحوه گسترش کارایی کرنل با بکارگیری پیمانه¬ها و وصله‌ها (patches)

- ملاحظات جداسازی[1] توسط لینوکس.
- جنبه‌های تعدد سکوها در لینوکس
- کاربردهای مهم لینوکس

این فصل چندین مفهوم فنی پیچیده را مورد بررسی قرار می‌دهد. طبیعت کرنلهای سیستم‌عامل به گونه‌ای است که بحث درباره آن را پیچیده می‌کند. هدف، آشنایی با این مباحث پیشرفته برای درک نحوهٔ تأثیر آنها بر تصمیم‌گیری در زمینه به کارگیری، مهاجرت (تغییر سکو یا سیستم‌عامل) و انجام پروژه‌ها است. همچنین اغلب مدیران فناوری اطلاعات نگران تکرار جنگهای کهن یونیکسی هستند که باعث ناهنجاریهایی در بازار شدند. افزایش درکتان از لینوکس و ساختار آن، در تصمیم‌گیریهای فناوری اطلاعات به شما نموده و از نگرانیهای شما می‌کاهد. خود را آماده کنید برای پاسخ به این سؤال که "چرا لینوکس متفاوت است؟"

در فصل پنجم به بررسی توزیعهای لینوکس خواهیم پرداخت. توزیعها ترکیبی هستند از کرنل لینوکس، سیستم گنو و اجزای مورد نیاز برای داشتن سیستمی کارآ. در برخی موارد یک توزیع، کارآیی مورد نیاز برای یک سیستم کامل را در بردارد و در برخی موارد نیز نیاز به تهیه برنامه‌های کاربردی اضافه است.

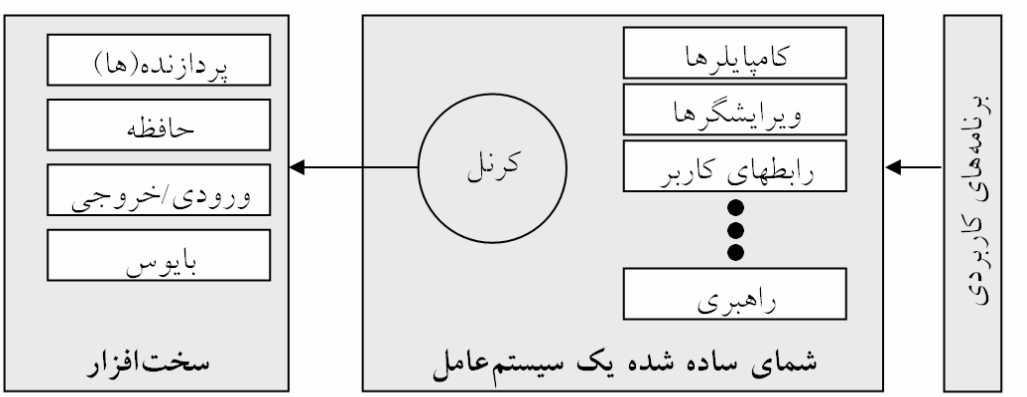
سیستم‌عامل

در زمینه آنچه که سیستم‌عامل باید دربرداشته باشد، استاندارد صددرصد پذیرفته شده‌ای وجود ندارد. زیرا اجزایی که در هر سیستم عرضه می‌شوند، در طول زمان به طور چشمگیری رشد و تغییر می‌کنند. در اواسط دهه ۸۰ میلادی– وقتی که بحث شبکه‌های کامپیوتری هنوز نوپا بود – شرکت-های زیادی پروتکلهای شبکه خود (مثل TCP/IP) را به همراه سیستم‌عاملهای محبوب می‌فروختند. با تکامل شبکه‌های کامپیوتری و تبدیل آنها به یکی از نیازهای اصلی برنامه‌های کاربردی، پروتکلهای شبکه جزء اصلی اکثر سیستم‌عاملها شدند.

---

[1] fragmentation

بحث اجزای تشکیل‌دهنده سیستم‌عامل به جایی رسیده است که بعضی عرضه‌کنندگان اجزای خاصی را ارائه می‌دهند و برخی دیگر نمی‌دهند. به هر حال، اغلب عرضه‌کنندگان، شکل ۱-۲ را به عنوان یک نمای ساده شده از مؤلفه‌های سیستم‌عامل مد نظر قرار داده‌اند. سیستم‌عامل لینوکس همه این اجزا را شامل می‌شود.



شکل ۲-۱: سیستم‌عامل

همان طور که می‌بینید، سیستم‌عامل، پل ارتباطی بین برنامه‌های کاربردی (که شما برای انجام کار خود به آنها نیاز دارید) و منابع سیستم (مثل بخشهای مختلف سخت‌افزاری) است. برای داشتن برنامه‌های کاربردی بیشتر و بالا بردن بهره‌وری سیستم، سیستم‌عامل باید توانایی انجام کار بیشتر را داشته باشد. سیستم‌عاملهای مصرف‌کنندگان، برنامه‌های کاربردی بسیاری (مثل پردازش تصویر، اجرای موسیقی و غیره) را به همراه دارند. سیستم‌عاملهایی که برای دنیای شرکتی طراحی شده‌اند، بر برنامه‌های کاربردی تمزکز کرده‌اند که قابلیت مدیریت کاربران زیاد را داشته باشد و هدف عمده آنها از این کار کنترل منابع کامپیوتر با حداقل هزینه است.

بهره‌گیری از منابع در لینوکس از سه جنبه قابل بررسی است:

۱.مدیریت، ۲.راهبری، ۳.کنترل.

از نقطه نظر سیستم‌عامل، مدیریت منابع دید جامعی از منابع موجود کامپیوتر و نحوه توسعه و به کارگیری آنها ارائه می‌دهد. مثلاً اگر شما یک خط هواپیمایی را مدیریت می‌کنید، هواپیمایتان را در مسیرهایی به کار می‌گیرید که درآمد بهینه متناسب با آن مسیر را به همراه داشته باشد. به طور

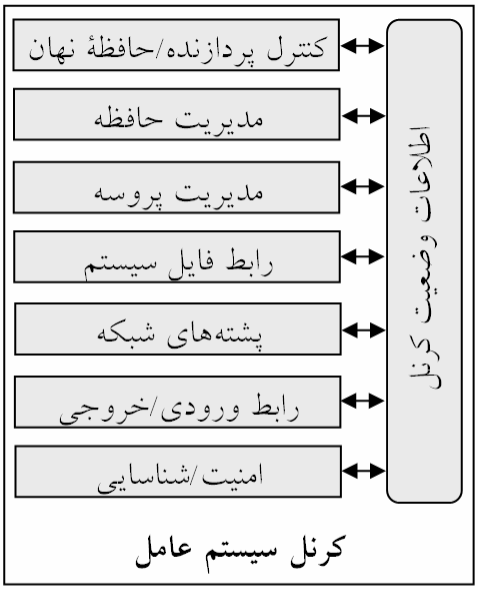
مشابه در لینوکس مؤلفه‌های مدیریت منابع به شما در مدیریت مؤلفه‌های سخت‌افزاری سیستم به طور یکپارچه کمک خواهند کرد.

پس از توسعه منابع کامپیوتر، ابزارهای سرپرستی منابع در لینوکس با کاراترین روش ممکن به پیکربندی منابع موجود کمک می‌کنند. در مورد مثال خط هواپیمایی، شما از برپایی هواپیما با مقدار سوخت مناسب، غذای خوب و مناسب برای مسافران و کارکنان مجرب اطمینان حاصل می‌کنید.

کنترل منابع در حوزه کرنل سیستم‌عامل لینوکس قرار می‌گیرد. کرنل منابع موجود را با توجه به طراحی و پیکربندی آنها کنترل می‌کند. درست مثل اینکه هواپیما با ارتفاع و سرعت از پیش تعیین شده پرواز می‌کند.

کرنل لینوکس

لینوکس از نظر فنی کرنل سیستم‌عامل است. شکل ۲-۲ مؤلفه‌های اصلی کرنل را مشخص می‌کند. برای درک بهتر، کرنل را می‌توان به موتور هواپیما تشبیه کرد. موتور، توان مورد نیاز هواپیما را تأمین می‌کند و این در حالی است که خلبان به طور مستقیم با موتور تعاملی ندارد. کرنل، توان سیستم‌عامل است که کاربران کامپیوتر هیچگاه مستقیماً با آن سروکار ندارند. در واقع می‌توان گفت کرنل نرم‌افزار عظیمی است که سخت افزار را کنترل می‌کند. کرنل کنترل و اختیار منابع سیستم مثل دیسکها، حافظه، واحد پردازش مرکزی (CPU) و غیره را در دست دارد. این وظیفه کرنل است که همه درخواستهای ورودی (از طرف برنامه‌های کاربردی و کاربران) برای منابع سیستم را مدیریت کند و پاسخگو باشد. کارایی کرنل در این زمینه به طراحی و پیکربندی آن بستگی دارد.

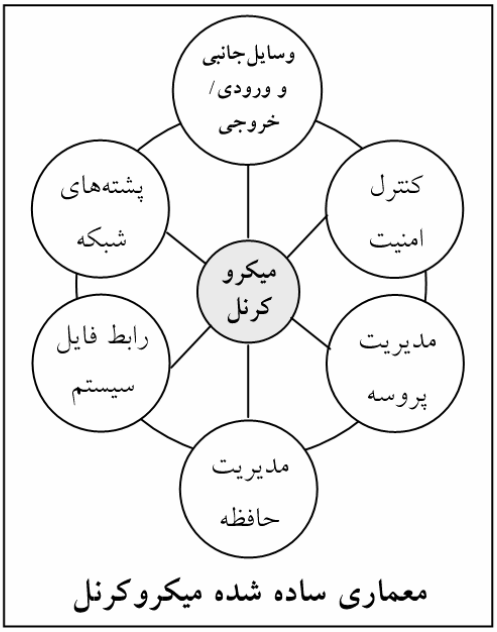


شکل ۲-۲: کرنل و عملکردهای سیستم عامل یکپارچه

طراحی کرنل

لینوکس یک کرنل یکپارچه است. همان طور که شکل ۲-۲ نشان می‌دهد، همه مؤلفه‌های کرنل را می‌توان یک برنامه کاربردی خاص و عظیم که روی سیستم اجرا می‌شود، در نظر گرفت. این برنامهٔ کاربردی ویژه دارای توانائیها و ویژگیهای خاص است که کار آن کنترل سیستم است. واژهٔ "برنامه کاربردی" نباید با برنامه‌های کاربردی -که برای انجام کارهای روزمره به کار می‌رود- اشتباه گرفته شود. هر اطلاعاتی در مورد حالت کرنل، در اختیار همه اجزای کرنل قرار می‌گیرد. نوع دیگری از کرنل که در سیستم‌عاملهای پیشرفته به کار می‌رود، میکروکرنل است. میکروکرنلها اکثر وظایف کنترلی اصلی را به چندین قسمت تقسیم می‌کنند. علاوه بر اینکه اطلاعات برای همه توابع قابل دسترسی است، پیامها از یک تابع به تابع دیگر منتقل می‌شوند. شکل ۲-۳ یک معماری ساده میکروکرنل را نشان می‌دهد.

طرفداران میکروکرنلها بر این باورند که پیمانه‌ای[1] بودن کرنل، کارآیی و قابلیت انتقال آن را افزایش می‌دهد. قابلیت انتقال به این معنی است که کل سیستم‌عامل را بتوان از روی یک لوح فشرده اجرا با خودتان جابه جا نموده و تنظیمات آن را منتقل کنید.



شکل ۲-۳: میکروکرنل

از سوی دیگر افرادی مثل لینوس توروالدز معتقدند که فرآیند انتقال پیام بین هر یک ازپیمانه‌های کنترل، پیچیدگی را افزایش داده و کرنل را مستعد کاستیها و نقاط ضعف مبهم می‌کند. ریچارد استالمن کرنلی برای سیستم گنو طراحی کرد که مبتنی بر معماری میکروکرنل بود و GNU Hurd نام داشت. ریچارد اعتراف می‌کند که پیچیدگی این کرنل، فرآیند توسعه و رفع اشکال آن را به طور قابل توجهی کند کرد. این درحالی است که کرنل یکپارچه توروالدز خیلی سریع برای پشتیبانی سیستم گنو قبل از آماده شدن GNU Hurd طراحی شد.

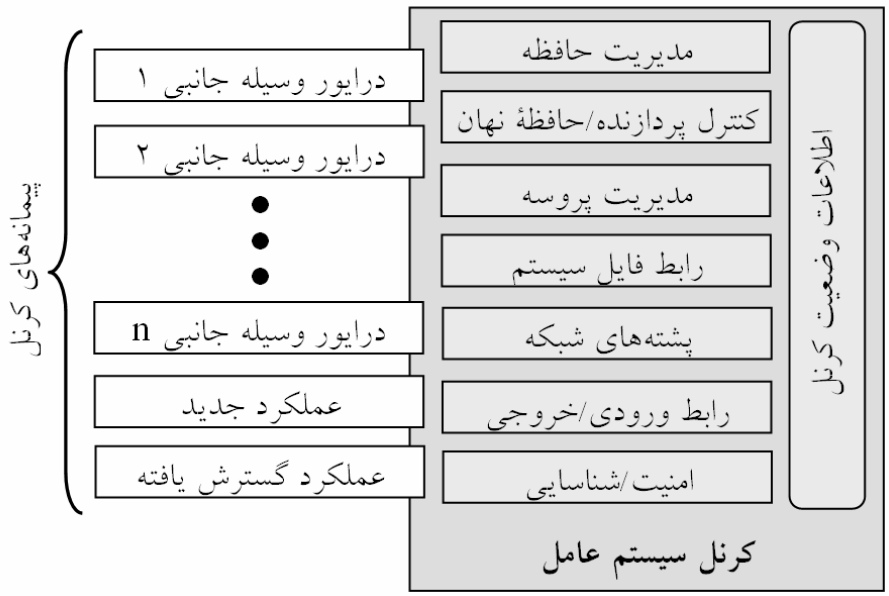
[1] (modular)

پیمانه‌های کرنل لینوکس

یکی از پیشرفتهای اصلی در نسخه ۲ کرنل لینوکس، استفاده از پیمانه‌ها بود. قبل از این، هر کارآیی یا قابلیت جدیدی که به کرنل اضافه می‌شد، باید در کرنل کامپایل شده و سیستم برای استفاده از آن قابلیت جدید دوباره راه‌اندازی می‌شد. پیمانه‌ها مکانیزمی برای افزایش کارایی کرنل لینوکس هنگامی که سیستم در حال کار است، فراهم می‌کنند. معمول‌ترین روش به کارگیری پیمانه‌ها از طریق درایورها است. یک درایور، نرم‌افزاری است که بین سیستم‌عامل و وسیله سخت‌افزاری ارتباط برقرار می‌کند. در قیاس با هواپیما درایورها مثل کنترل جریان بنزین، راهبری هواپیما و مکانیزم‌های دیگری هستند که خلبان برای کنترل هواپیما به کار می‌گیرد. خلبانان بدون نگرانی از مشخصات موتور این کنترلها را بکار می‌گیرند. درایورها نیاز به داشتن اطلاعات در مورد وضعیت فعلی کرنل دارند و همچنین باید از طریق اجزای کرنل کنترل شوند.

اغلب درایورها توسط اشخاصی از انجمنهای مختلف توسعه متن‌باز نوشته می‌شوند. توجه داشته باشید که هنگام ساخت کرنل، مشخص‌کردن همه دستگاههای مورد استفاده و پیکربندیهای سیستم که مورد نیاز کاربر نهایی خواهند بود، تقریباً غیرممکن است. بدون پیمانه‌ها هر درایوری باید درون کرنل کامپایل شود که این امر اندازه کرنل را به طور چشمگیری افزایش می‌دهد و کارایی سیستم را پایین می‌آورد. شکل ۲-۴ پیمانه‌های کرنل لینوکس را نمایش می‌دهد.

در دنیای متن‌باز همه قادر به تغییر یا توسعه کارایی کرنل لینوکس، از طریق به کارگیری پیمانه‌ها هستند.



شکل ۲-۴: پیمانه‌های کرنل لینوکس

وصله‌های[1] کرنل لینوکس

همان‌گونه که پیمانه‌ها مکانیزمی برای افزودن کاراییهای جدید به کرنل لینوکس فراهم می‌کنند، وصله‌ها نیز راهی برای گسترش یا تغییر کرنل ارائه می‌دهند.

وصله‌ها را می‌توان مکانیزمی برای ترمیم و بازسازی کاستیها در نظر گرفت. در مورد لینوکس وصله‌ها برای تغییر دادن یا اضافه کردن کارآیی به کرنل نیز به کار می‌روند. مجدداً در قیاس با هواپیما که کنترلهای راهبری و کنترل جریان بنزین، معادل درایورها بودند، وصله‌ها در مواردی موتور را تعمیر می‌کنند و در مواردی راه‌هایی برای پیاده‌سازی عملکردهای جدید مثل اضافه کردن قابلیت خنک‌سازی به موتور را فراهم می‌کنند.

در حین گسترش لینوکس در سازمان خود متوجه می‌شوید که برنامه‌های کاربردی جدیدی برای کار روی لینوکس نوشته می‌شود. باید در زمینهٔ وابستگیها[2] چه در مورد نسخهٔ کرنل و چه در مورد وصله‌ها و پیمانه‌هایی که احتیاج به نصب دارند، دقت لازم را لحاظ کنید.

---

[1] patch

[2] dependency

تکثر کرنل

یکی از بزرگترین نگرانیهای تازه‌واردها به دنیای لینوکس، مسأله تکثر است. تکثر به معنی پیاده-سازی‌های گوناگون و پراکنده از کرنل است.

این مسأله از جنگهای تاریخی یونیکس نشأت می‌گیرد. در اصل یونیکس به منظور فراهم کردن سیستمی با مجموعه‌ای از واسطهای برنامه‌نویسی رایج شکل گرفت. برای انتقال برنامه‌های کاربردی از یک سیستم یونیکس به سیستمی دیگر تنها نیاز به کامپایل مجدد آن روی سیستم مقصد بود. به هر حال عرضه‌کنندگان یونیکس نیاز به محصولات متنوع داشتند. بنابراین هر عرضه‌کننده، کارآیی جدیدی به سیستم‌عامل اضافه می‌کرد و یا کارآیی موجود را توسعه می‌داد که در اکثر موارد این عمل به روشهای متفاوتی صورت می‌گرفت. نتیجه این شد که با اینکه سیستمهای یونیکس ریشه و عملکرد مشترکی داشتند، ولی یونیکسهایی که عرضه می‌شدند آنقدر متفاوت بودند که کاربر برای انتقال برنامه‌های کاربردی از یک نسخه به نسخه دیگر، به دردسر می‌افتاد.

این تفاوتها باعث به تله افتاندن[1] کاربران نیز می‌شد و هزینه‌ها را هنگام تعویض عرضه‌کننده بالا می‌برد. تولیدکنندگان نرم‌افزار نیز با فشار کاری ناشی از تفاوت این محصولات مواجه بودند و نیاز داشتند هزینه‌های انتقال و پشتیبانی هر سکوی کامپیوتری جدید را ضمن دراختیار گرفتن بیشترین سهم بازار، تعدیل کنند. هر سکوی پشتیبانی‌شده به پیکربندی و آزمایش گسترده‌ای نیاز دارد. بنابراین تکثر کرنل هزینه‌ها را هم برای تولیدکنندگان نرم‌افزار و هم برای مشتریان به طور چشمگیری افزایش می‌دهد.

زمانی که شرکتها و تولیدکنندگان نرم‌افزار کار خود را با لینوکس آغاز کردند و نام تعداد زیاد توزیعهای تجاری لینوکس به گوش آنها رسید، ترس از جنگهای یونیکسی آنها را فرا گرفت. به هر حال همه توزیعهای لینوکس در کرنل لینوکس مشترک هستند و این مسأله‌ای است که لینوکس را متفاوت می‌سازد. کرنل مشترک لینوکس از نگرانیهای مربوط به هزینه‌های انتقال و پشتیبانی

---

[1] Lock-in

می‌کاهد و در مقایسه با مسایلی که تولیدکنندگان نرم‌افزار در گذشته هنگام کار با یونیکس مواجه می‌شدند، برای آنها تسهیلات بیشتری از نظر هزینه و عرضه در بازار به همراه دارد.

اگر چه همه عرضه‌کنندگان با یک کرنل مشترک کار می‌کنند، وصله‌ها و خصوصیات دیگر می‌توانند تفاوتهایی در میان توزیعهای لینوکس بوجود آورند. فصل هفتم شما را با استانداردهای وضع شده در این زمینه که این اختلافات راجبران می‌کند، آشنا خواهد نمود.

کرنل لینوکس متن‌باز است. این حقیقت، از انگیزهٔ عرضه‌کنندگان برای ارائه محصولات مختلف که در سطح پائین زنجیرهٔ ارزش قرار دارند، می‌کاهد. در فصل پنجم وقتی توزیعهای لینوکس مفصلاً شرح داده می‌شوند، نگاه اجمالی به زنجیرهٔ ارزش و نحوه تفاوت ارزش محصولات، خواهیم داشت.

زیبایی متن‌باز در این است که اکنون عرضه‌کنندگان انگیزه بالایی برای همکاری و مساعدت با جامعه متن‌باز برای گسترش دادن کارآیی کل سیستم، جهت دستیابی به اهداف بالای خود، دارند. عرضه‌کنندگان نرم‌افزار در عین اینکه هزینه‌های توسعه را کاهش می‌دهند، در جلب رضایت مشتریان می‌کوشند.

تولیدکنندگان می‌توانند از یک زیربنای مشترک برای توسعه و ایجاد برنامه‌های کاربردی خود استفاده کنند. درنهایت شرکتها بجای ارائه کارهایی دارای عملکردهای سطح پائین و انحصاری، می‌توانند از سیستم مشترک بین عرضه‌کنندگان سود برند و انرژی خود را بر سایر نیازهای تجاری خویش متمرکز کنند.

گرچه متن‌باز بودن لینوکس، پراکندگی کرنل را غیر محتمل می‌سازد، شرکتها و تولیدکنندگان نیاز به مدیریت برخی پیچیدگیها در محیط لینوکس دارند. همان طور که هر سیستمی توسعه می‌یابد، کرنل نیز گسترش یافته است و اکنون چندین نسخه از آن موجود است. تولیدکنندگان نرم‌افزار، توسعه‌دهندگان غیررسمی و دیگر ارائه‌دهندگان نرم‌افزارها وصله‌هایی برای کرنل طراحی می‌کنند و لازم است در زمینه مدیریت وابستگیها چاره‌ای اندیشیده شود. بنابراین گونه‌های زیادی از پیمانه‌های کرنل (بصورت درایورها) وجود دارد که ممکن است حین استفاده از لینوکس به آنها

احتیاج داشته باشید. نهایتاً عرضه‌کنندگان توزیع ترکیب خاصی از نسخه‌های کرنل، بسته‌های نرم‌افزاری و بهبودهای صورت گرفته روی آنها را ایجاد کنند. بعداً در فصلهای ۵ و۶و۷ به توزیعها، استانداردها و فرایندهای توسعه‌ای که امکان مدیریت این پیچیدگی را می‌دهند، خواهیم پرداخت.

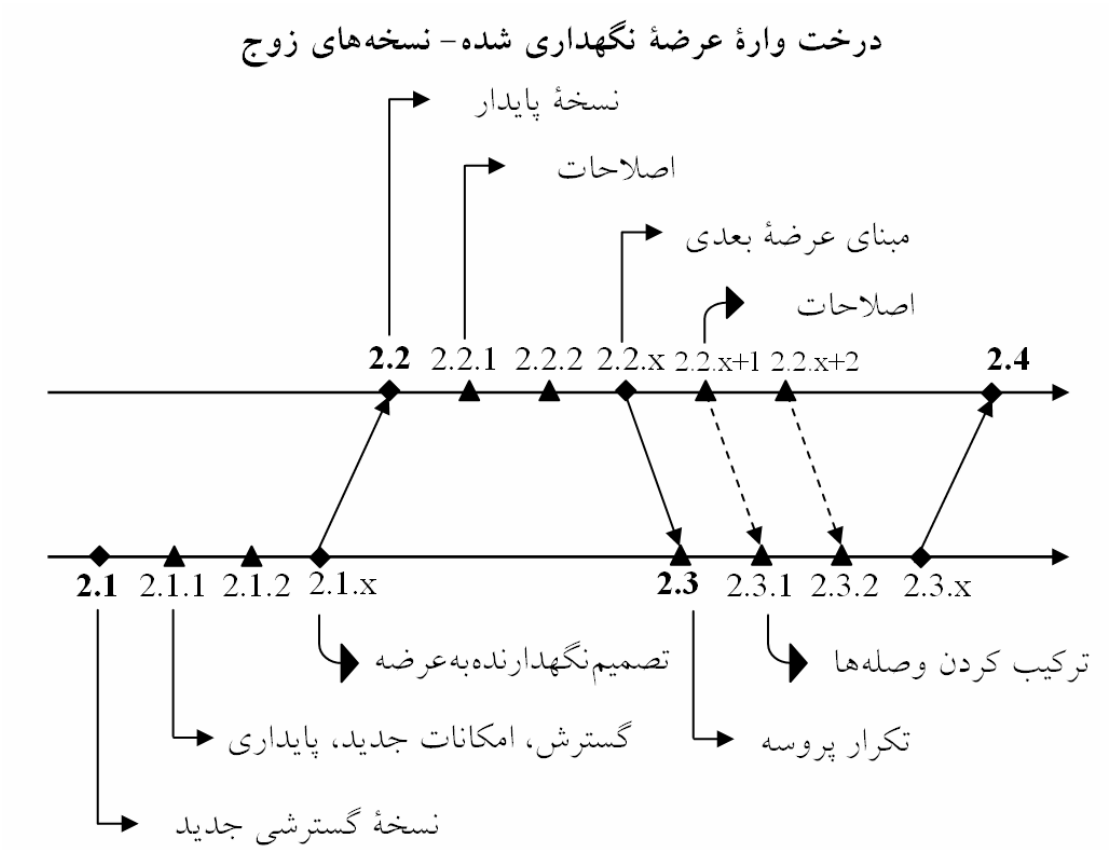
طراحی کرنل لینوکس و کنترل نسخه‌ها

لینوس توروالدز مالک نشان تجاری ثبت شده لینوکس است. به عبارت دیگر او طراحی نسخه‌های جدید لینوکس را تأیید و تصویب می‌کند. به هر حال مرکز جهانی توسعه کرنل لینوکس، Kernel.org است که یک فرآیند توسعه باز و مبتنی بر همکاری را هدایت می‌کند.

به نسخه‌های جدید کرنل عدد فردی به عنوان عدد زیر نسخه اختصاص می‌دهند. نسخه‌های کرنل ۲۰۱.x ، ۲۰۳.x و ۲۰۵.x همه کرنلهایی هستند که طراحی شده و یا در حال طراحی هستند که می‌توانید آنها را مانند نرم‌افزار ناپایدار کیفیت سنجی آلفا و بتا در نظر بگیرید. قاعدتاً هیچ شرکتی محصول تولیدی خود را بر مبنای این کرنلها گسترش نمی‌دهد. نسخه‌های کرنل با زیر شماره‌های زوج x. ۲.۰ ، ۲۰۲.x و ۲۰۴.x نیز کرنلهای طراحی شده‌ای هستند که آماده استفاده‌اند. شکل ۲ـ۵ این سیستم نسخه‌گذاری را نمایش می‌دهد.

یکی از نگرانیهای دنیای لینوکس، نداشتن طرح از پیش تعریف شده‌ای برای آینده و همچنین نداشتن تاریخ قطعی عرضه[1] است. در جهان توسعهٔ سنتی، طرحهای از پیش تعیین شده، کارآییهای آیندهٔ محصول را در هر زمان مشخص می‌کنند. این حقیقت که طرح خاصی برای لینوکس موجود نیست، باعث نگرانی بسیاری از مدیران می‌شود.

[1] Committed release dates



شکل ۲-۵: چرخهٔ توسعهٔ کرنل لینوکس

بعد از این که کرنل طراحی و تصویب شد، لینوس توروالدز چرخه توسعه جدیدی را آغاز می‌کند (زیرشماره فرد بعدی اعلام می‌شود). در این هنگام انجمن کرنل برای تعیین لیستی از ویژگیهای که نسخه بعدی کرنل گردهم خواهند آمد.

این انجمن تصمیم‌گیرندگان، در عین خبره بودن، به خواسته‌های مشتری نیز اهمیت می‌دهند. بزرگترین تشویق برای یک برنامه‌نویس این است که شاهد به کارگیری گسترده نرم‌افزار خود باشد. مرحله دیگر عرضه محصول در زمانی است که انجمن آن را آماده تشخیص دهد، نه وقتی که واحدهای بازاریابی یا مدیران غیرمرتبط تشخیص دهند.

اغلب شرکتها، نرم‌افزار را به واسطهٔ نیاز بازار، فشارهای رقابتی و خواسته‌های مشتری طراحی می‌کنند. بنابراین سازمانهای مهندسی بخاطر الزام به تحویل کار در زمان مقرر، مجبور به استفاده از معماریها و طراحیهای نه چندان مناسب می‌شوند. یکی از ضعفهای این مدل این است که در اکثر

موارد، محاسبات و تلاشهای طراحی و معماری، با به پایان رسیدن زمان توسعه به اتمام می‌رسد و این از کیفیت و کارآیی سیستم می‌کاهد. از آن جایی که هدف جامعه متن‌باز، ارائه یک سیستم با کیفیت است و تأکیدی بر بازگشت سرمایه‌های اقتصادی وجود ندارد، متناظراً این چرخه مخرب توسعه هم شکل نمی‌گیرد. این مسأله به این معنی نیست که انجمنها نسبت به نیازهای بازار و درخواستهای مشتریان بی‌توجه هستند، بلکه منظور این است که فرآیندهای طراحی، عقلانی‌تر و با روند منطقی‌تری شکل می‌گیرند که از اثرات منفی به دور است. این چرخه طراحی نرم‌افزار، عمده-ترین دلیل این است که پروژه‌های متن‌باز با سرعت بیشتری نسبت به همتایان تجاری خود رشد می‌کنند و پایداری بیشتری دارند.

بهترین راه برای شتاب دادن طراحی کرنل جدید لینوکس و یا اثرگذاری بر مجموعه مشخصه لینوکس خیلی ساده است: خود را درگیر کنید! انجمن توسعه لینوکس بصورت ایجاد و حفظ ارتباط و تعامل کار می‌کند. با آنها همکاری کنید و در تیم توسعه لینوکس شرکت کنید. این یک معامله برنده (پایاپای) برای شما و انجمن کرنل است. قبول مسئولیتهای کوچک و پشت سرگذاشتن مراحل اولیه، شما را از نزدیک با نحوه پیشرفت فرآیند طراحی و توسعه و زمان احتمالی عرضه نسخه‌های جدید آشنا می‌کند. این همکاری به طراحی بهتر سازمان و زیرساخت فناوری اطلاعات آن منجر خواهد شد. اگر چه لینوکس و دیگر پروژه‌های متن‌باز طرح و قالب منتشر شده‌ای ندارند، اما طبیعت توسعه باز، دید روشنی از فرآیند توسعه در اختیار شما قرار می‌دهد. این دید وسیع باعث می‌شود بتوانید تحلیل بهتر و نزدیک به واقع بینانه‌تری از آینده، نسبت به طراحیهای از پیش چاپ شده، بدست آورید.

از آنجایی که کرنلهای توسعه داده شده به ندرت در محیط تولید تست می‌شوند، مشخص کردن همهٔ کاستیهای سیستم ممکن نیست. گرچه جامعه متن‌باز بهترین آشکارکنندهٔ کاستیهاست، در عمل استفاده از یک محیط تولید مشارکتی بهترین راه برای زیروروکردن کاستیهای یک سیستم است.

این فرآیند ممکن است خطاهایی را آشکار کند که نیاز به تصحیح دارند. وقتی لینوس توروالدز یک نسخهٔ نامگذاری شده با شماره‌های زوج کرنل لینوکس را طراحی می‌کند، مسئولیت نگهداری آن به "توساتی"[1] سپرده می‌شود (این مسئولیت را چندین سال آلن کوکس برعهده داشت). همانطور که شرکتها لینوکس را گسترش می‌دهند و کاستیها را آشکار می‌سازند، "مارچلو" تغییرات لازم را روی کرنل اعمال می‌کند. لینوس و مارچلو با هم در ارتباط مستمر هستند و این خیلی مهم است که مارچلو تغییراتی را که از لحاظ معماری اثرات منفی بلندمدت روی کرنل می‌گذارند، پیاده‌سازی نکند. همین طور هم هست و قرارداد و اطمینان بین لینوس و مارچلو و آمادگی آنها برای پذیرفتن پیشنهادات از طرف انجمن توسعه متن‌باز، به گونه‌ای است که تحقق این اتفاق بعید به نظر می‌رسد. وقتی شما با شرایط ناخواسته در گسترش لینوکس مواجه می‌شوید، عملی‌ترین راه برای مقابله با آنها، ایجاد یک رابطه قوی با انجمن توسعه‌دهندگان کرنل است. اغلب افرادی که با این کاستیها روبرو می‌شوند، نه تنها مشکل را مستند می‌کنند، بلکه آنرا به صورت دیگری مثل نیز درخواست اصلاح کد، به مارچلو توساتی ارائه می‌دهند.

مطمئناً شرکتهای زیادی وجود دارند که کارمندان فناوری اطلاعات آنها از تعداد، تجربه و دانش لازم برای کارشناسی کرنل برخوردار نیستند. در این مواقع شرکتها معمولاً از عرضه‌کنندگان لینوکس برای پشتیبانی کمک می‌گیرند. ما این مدلهای پشتیبانی را با تفصیل بیشتر در فصل هشتم بررسی خواهیم کرد.

وقتی طراحی کرنل به حد قابل قبولی می‌رسد، این کرنل پایدار بعنوان پایه‌ای برای توسعه جدید کرنل به کار خواهد رفت و این چرخه تکرار خواهد شد. وقتی که یک کرنل تولیدی و یک کرنل توسعه‌ای هر دو در حال گسترش هستند، مارچلو و لینوس برای اطمینان از اینکه هم گامی آنها حفظ می‌شود، باهم کار می‌کنند.

---

[1] Tosatti

پشتیبانی چند سکو

یکی از نقاط قوت لینوکس از این واقعیت نشأت می‌گیرد که لینوکس گسترده‌ترین سیستم‌عامل قابل انتقال به سکوهای دیگر[1] است. لینوس توروالدز در ابتدا کرنل را به عنوان پروژه‌ای برای مدرسه طراحی کرد. چرا که او سیستم‌عامل شبه یونیکسی، برای کامپیوتر شخصی مبتنی بر اینتل ۳۸۶ جدیدش می‌خواست.قابلیت انتقال هدف وی نبود. کرنل به طور مستقیم با ریزپردازنده ۳۸۶ مرتبط بود. همانند آنچه در مورد اکثر پروژه‌های متن‌باز رخ می‌دهد، کمتر کسی می‌توانست روند رشد این پروژه را پیش‌بینی کند.

توسعه‌دهندگان زیادی به قابل‌انتقال‌کردن کرنل لینوکس روی آوردند و سعی بر ارائه ساختار مناسب داشتند. امروزه درخت توسعه کرنل لینوکس هم انشعابات مستقل از معماری و هم انشعابات وابسته به معماری دارد. انشعاب مستقل از معماری و کد مربوط به معماری اینتل x86 همگی بوسیله لینوس نگهداری می‌شوند (مؤلفه‌هایی مثل درایورها و ریزپردازنده‌های دیگر توسط گروه جداگانه‌ای از توسعه‌دهندگان نگهداری می‌شوند.) وقتی این توسعه‌دهندگان با مشکلات مربوط به پردازنده مواجه می‌شوند، می‌توانند تغییرات لازم را برای اطمینان از کار کردن صحیح سیستم اعمال کنند. در برخی موارد نیاز است که تغییرات به مؤلفه‌های مستقل از معماری اعمال شود که به تبع آن باید تغییراتی به قسمت تحت کنترل لینوس نیز وارد شود. نگهدارندگان کرنل برای طراحی و اعمال این تغییرات در رابطهٔ تنگاتنگی با لینوس کار می‌کنند.

بازار بزرگ در مقابل پردازنده‌های نامشهور

شرکتهایی که توسعه لینوکس را از طریق ریزپردازنده‌های نامشهور دنبال می‌کنند، باید فاکتورهای اضافه‌ای در تصمیم‌گیریهای خود لحاظ کنند. به یاد آورید که یکی از مزایای اصلی لینوکس این است که لینوکس اقتصاد کالایی (عمومی شده) را به سیستم‌عاملها تعمیم می‌دهد. الحاق ریز-پردازنده‌های کالایی (عمومی شده) به بازار لینوکس آن را به گزینه‌ای مقرون به صرفه و باارزش برای تجارت بدل می سازد.

---

[1] portable

عده زیادی قابل انتقال بودن لینوکس را یکی از توانمندیهای لینوکس می‌دانند. در مواردی ممکن است به کارگیری لینوکس در پردازنده‌های اختصاصی گزینه مطلوب و ارزشمندی باشند. مثلا وقتی خصوصیات، کارآیی یا مجموعه ویژگیهای خاصی مدنظر باشند که پردازندهٔ خاصی از عهده آنها بهتر برآید. بکارگیری لینوکس تحت ریزپردازنده‌های نامشهور نیز ایدهٔ ارزشمندی است بخصوص وقتی که شما نیاز به انتقال از یک سکو یا سیستم‌عاملی دارید و می‌خواهید از پردازنده‌های اختصاصی به پردازنده‌های مشهور روی آورید. بعضی از عرضه‌کنندگان توزیعهای لینوکس، محصولات لینوکس خود را روی پردازنده‌های نامشهور تولید نموده که این فرآیند را تسهیل می‌کند. هنگام ارزیابی پردازنده‌های جایگزین موارد زیر را لحاظ کنید:

- هم اکنون توسعه اصلی لینوکس روی معماری x86 انجام می‌شود.
- برنامه‌های کاربردی مورد نیاز برای کار شما ممکن است موجود نباشند و در مدت کوتاهی حاضر شوند.
- تولیدکنندگان نرم‌افزار ممکن است از پشتیبانی برنامه‌های کاربردی خود روی هر سیستم یا پردازنده غیر متداول سرباز زنند. آنها ممکن است به منابع لازم برای پشتیبانی از معماری‌های دیگر دسترسی نداشته باشند.
- جامعه توسعه‌دهندگان متن‌باز به ندرت به منابع مورد نیاز برای توسعه و تست برنامه‌های خود روی معماریهای دیگر دسترسی دارند.
- تغییراتی که لازم است در اجزای وابسته به معماری کرنل اعمال شود، ممکن است آن را برای افراد دیگر نامطلوب سازد.
- اکثر نویسندگان درایورها امکان تست کردن دستگاه‌های خود را روی معماریهای دیگر ندارند.
- آموزش‌دیدگان لینوکس، به ندرت با لینوکس روی معماریهای دیگر آشنایی دارند.

گرچه این لیست فهرست مهمی از نکاتی است که باید مدنظر قرار داد، با این حال ممکن است شما قصد گسترش روی پردازنده‌های نامشهور را داشته باشید. امروزه کرنل لینوکس علاوه بر معماری پردازنده x86، بر روی معماریهای زیر نیز موجود می‌باشد:

- Opteron از AMD

- PA-RISC و Alpha از HP
- PowerPC از IBM
- S/390 از IBM
- خانواده Itanium Processor از Intel
- 68K از Motorola
- SPARC از SUN

میزان پشتیبانی در هر مورد متفاوت است. در اکثر موارد توزیعهای کامل لینوکس (فصل ۵) موجود هستند. اطلاعات بیشتر در مورد پشتیبانی لینوکس از یک خانواده خاص از پردازنده‌ها در بخش مرجع پایان کتاب آمده است.

لینوکس روی میزکار

در حال حاضر گسترش لینوکس روی میز کار، مورد بحث محافل زیادی است. آمارها نشان می‌دهد لینوکس روی میز کار نفوذی کمتر از ۲ درصد و به گفته بعضی آمارها کمتر از نیم درصد داشته است. طرفداران لینوکس ادعا می‌کنند که لینوکس برای مطرح شدن به عنوان سیستم‌عامل رومیزی ارزشمند، نیاز به زمان دارد.

مهمترین اثبات این ادعا، شهرت لینوکس در اغلب کشورهای جهان است. در دسترس بودن برنامه-های کاربردی، کاربرپسند بودن، قابلیت جابجایی از یک سکو یا سیستم‌عامل به دیگری و هزینه‌های آموزش مجدد، گسترش لینوکس روی میزکار بسیاری از سازمانها را با مشکل مواجه کرده است.

بخشهای کلیدی از بازار وجود دارند که لینوکس می‌تواند گزینهٔ مناسبی برای میز کار باشد. شرکتهای برنامه‌نویسی که قبلا از ایستگاه‌های کاری یونیکس استفاده می‌کردند، جایگاه مناسبی برای میزکارهای لینوکس هستند. همچنین برنامه‌های کاربردی تصویرنگاری پیشرفته مثل صنعت بازیها و سرگرمیها و تصویربرداری پزشکی نیز کاندیداهای مناسبی برای گسترش لینوکس هستند.

توسعه‌دهندگان نرم‌افزار نیز جذب لینوکس رومیزی شده‌اند. زیرا آنها می‌توانند با استفاده از کامپیوترهای شخصی کم هزینه لینوکس، برنامه‌های کاربردی را برای محیطهای یونیکس گسترش دهند. در نهایت ایستگاه‌های کاری مالی که برای کاربردهای خاص مثل خودپردازها استفاده می‌شوند، می‌توانند رویکردهای مناسبی برای میزکار لینوکس باشند. همه موارد فوق پیشینه مشترکی در استفاده از ایستگاه‌های کاری یونیکس دارند. درحقیقت لینوکس می‌تواند با کمک یک کامپیوتر شخصی ارزان قیمت، تجربه یونیکس را در اختیار کاربران پیشرفته قرار دهد.

انجمن متن‌باز دو واسط کاربری رومیزی ارایه داده است: محیط رومیزی GNOME و محیط رومیزی K (یا KDE). آنها را می‌توانید مانند داشبورد ماشین خود فرض کنید.

اگر با میزکارهای یونیکس آشنا باشید می‌توانید GNOME و KDE را معادل CDE در یونیکس در نظر بگیرید. هر محیط رومیزی، «حس و ظاهر» یکتای خودش را دارد و واسط برنامه‌نویسی خاصی را پشتیبانی می‌کند. این واسط برای CDE، Motif، برای GNOME، GTK و برای KDE، Qt است. یکی از زیباییهای لینوکس این است که صرفه‌نظر از محیط رومیزی که انتخاب می‌کنید، می‌توانید برنامه‌های کاربردی خود را روی هریک از این واسطهای برنامه‌نویسی اجرا کنید.

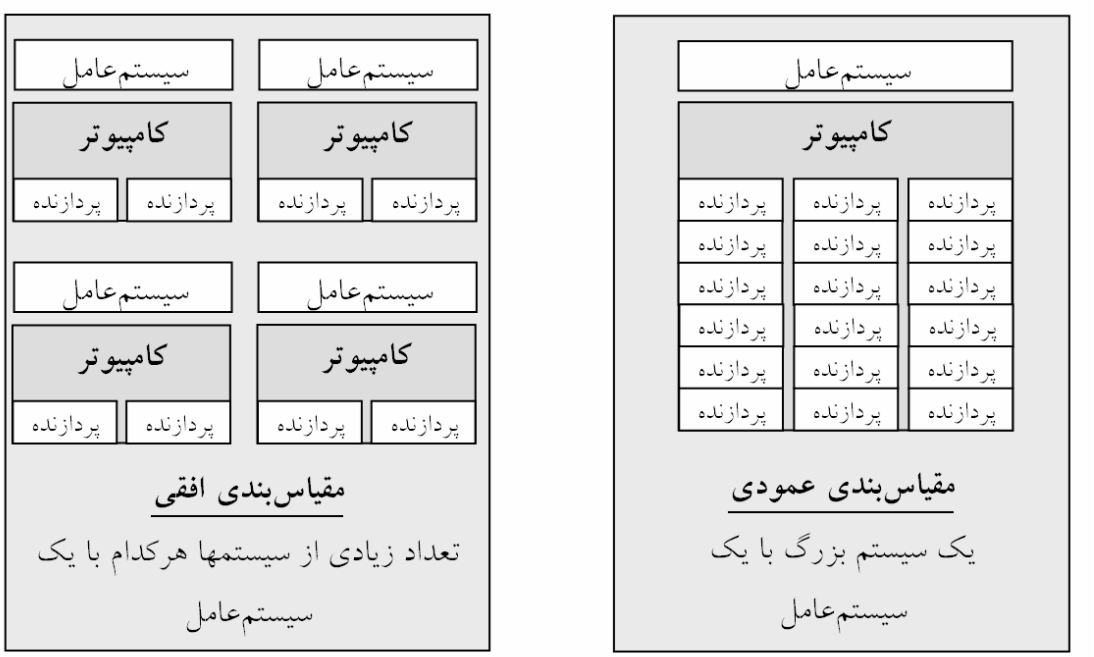
شرکت شما می‌تواند یک محیط رومیزی خاص را انتخاب کند و یا اینکه به کاربران خود اجازه انتخاب دهد و آنها را از اجرا شدن برنامه‌های کاربردی خود مطمئن سازد.

تعدادی برنامهٔ کاربردی سودمند اداری برای میزکارهای لینوکس موجود است. مشهورترین آنها نرم-افزاری Star Office می‌باشد که توسط SUN ارائه شده است و Open Office که نسخه متن‌باز Star Office است و توسط SUN و جمعی از توسعه‌دهندگان متن‌باز نگهداری می‌شود.

مقیاس پذیری عمودی و افقی

کرنل ۲،۴ گام بلندی برای مقیاس‌پذیری چند پردازنده‌ای (یا عمودی) بود. بسته به کاربرد، لینوکس می‌تواند به خوبی تغییر اندازه دهد و گروه‌های صنعتی و کاری با این روند به جلو می‌روند. کرنل

لینوکس نوعاً برای کار تحت یک پیکربندی چندپردازنده و یا تک پردازنده کامپایل می‌شود. پیکربندی چندپردازنده مقیاس‌پذیری عمودی را ممکن می‌سازد. قبل از شکل‌گیری کرنل 2.4 تنها روش مقیاس‌پذیری، گروهی (یا افقی) بود. یکی از فناوریهای گروهی درحال توسعه از سال ۱۹۹۴ ، Beowulf می‌باشد. شکل ۲-۶ تفاوت بین مقیاس‌پذیری عمودی و افقی را نشان می‌دهد. برنامه‌های کاربردی زیادی که مبتنی بر پردازش داده سنتی بودند، اکنون کرنل و سیستم‌عاملی دارند که مقیاس‌دهی عمودی را برای دربرگرفتن طیف وسیعی از نیازمندیها فراهم می‌سازد. مواردی که به پردازش موازی حجیم نیاز داشتند نیز بیشتر از ۸ سال است که صاحب این فناوری شده‌اند.



شکل ۲-۶: مقیاس‌بندی افقی و عمودی

پروژه مدرسه‌ای لینوس توروالدز، اکنون در مدت زمان کوتاهی به سطحی از کاربرد و پیچیدگی - معادل با آنچه توسط سیستمهای تجاری یونیکس عرضه شده بود- رسیده است.

لینوکس توکار

شرکتهای بسیاری وجود دارند که در ارایه یک لینوکس برای کاربردهای توکار (مانند کنترل اتوموبیل) تخصص دارند. کرنل لینوکس قابلیت خوبی برای برنامه‌های کاربردی توکار دارد. مثلاً

می‌توان یک کرنل لینوکس با کارآیی محدود و مقیاس کوچک ساخت که با یک حافظه یک مگابایتی کار کند.

با توجه به مجوز لینوکس، می‌توان آنرا به طور رایگان کپی برداری کرد. از آنجایی که هر دستگاه تولیدی نیازمند یک برنامه کاربردی توکار است، استفاده از لینوکس باعث می‌شود میلیونها دستگاه تولید شود، بدون اینکه نیازی به پرداخت حق الامتیاز باشد.

همچنین شما می‌توانید از کار شرکتهای دیگر به ویژه آنهایی که در زمینه کرنلهای توکار لینوکس کار می‌کنند، استفاده نمائید. اگر شما نیازمندیهای توکار پیچیده یا خاص دارید، می‌توانید یک کرنل توکار دلخواه خود طراحی نموده یا از خدمات شرکتهای دیگر استفاده کنید. تصمیم‌گیری در این زمینه به تواناییهای شما در پشتیبانی از محیط مورد نظر بستگی دارد.

جمع بندی

شما در این فصل یکی از پیچیده‌ترین موضوعات در علم کامپیوتر را بررسی کردید. درک کرنل لینوکس یکی از پایه‌های کلیدی برای فهم نحوه شکل گرفتن یک سیستم لینوکس کامل و چگونگی گسترش آن در سازمان است.

وقتی در فصل ۵ توزیعهای لینوکس بررسی می‌شوند، خواهید دید که کرنل چگونه به سیستم بزرگتر متصل می‌شود.

در فصل بعد، سعی خواهیم کرد درک شما را از قانون مالکیت معنوی گسترش دهیم و نمونه جدیدی در فرآیندهای توسعه نرم‌افزار را بررسی کنیم. متن‌باز بودن یکی از دلایلی است که توسعه لینوکس و برنامه‌های کاربردی دیگر را ممکن می‌سازد و فهم نحوه کار پروژه‌های متن‌باز و شالوده قانونی آن پیش نیاز توسعه لینوکس است.

فصل سوم:

متن‌باز- هدایت مسیر قانونی به آزاد بودن

پیشرفت سریع لینوکس سبب شده است که بتواند جایگاه ویژه‌ای در صنعت پیدا کند. لینوکس تنها پروژه متن‌باز عظیمی است که صدها توسعه‌دهنده در ارتباط با آن کار می‌کنند و قابلیتهای متن‌باز را به خوبی به نمایش گذاشته است. بنابراین شما به عنوان مدیر فناوری باید در زمینه متن‌باز خبره باشید. این متدلوژی جدید توسعه نرم‌افزار و مدل مجوزدهی آن، جدیدترین فرصتها و بزرگترین تهدیدها را برای سازمان شما به همراه دارد. لینوکس فرصتهای جدیدی فراهم می‌کند، زیرا راه‌های جدیدی برای صرف هزینه‌های کمتر و ایجاد مزایای رقابتی پیش رو می‌نهد و تهدیدهایی به دنبال دارد. چرا که حرفه و تجارت را درگیر یک سری ریسک می‌کند. از آنجائیکه توسعه‌دهندگان زیادی در سازمان از منابع نرم‌افزاری متن‌باز استفاده می‌کنند، شما باید در این باره توجه و دقت کافی داشته باشید. مشکل دیگر اینجاست که همین توسعه‌دهندگان، بطور کامل به آفات متن‌باز اشراف ندارند. شما می‌توانید با خبره شدن در این زمینه، در حالی که از مزایای فرصتهای پیش رو بهره می‌برید، ریسکها را حذف کنید. اهداف این فصل عبارتند از بررسی موارد زیر:

- آزادی بودن و متن‌باز
- تعریف متن‌باز (OSD)
- مجوزهای متن‌باز شامل GPL
- مجوزهای غیر متن‌باز که اغلب با مجوزهای متن‌باز اشتباه گرفته می‌شوند
- متدولوژی توسعه متن‌باز

این فصل بر تشریح متن‌باز و حرکت قانونی آن تمرکز دارد. قسمت سوم این کتاب بحثهای کاملی روی فرآیندهای تجاری مرتبط با استفاده و متن‌باز شرکت نموده و راهنمائیهایی ارائه می‌دهد که به شما کمک می‌کند بتوانید درباره زمان پیاده‌سازی نرم‌افزار متن‌بازی که توسعه داده‌اید، تصمیم بگیرید.

آزادی متن‌باز بودن

ریشه متن‌باز از نرم‌افزار آزاد[1] گرفته شده است. اکثر مردم وقتی برای اولین بار واژه نرم‌افزار آزاد را می‌شنوند، فکر می‌کنند این واژه به معنای نرم‌افزار رایگان است. این تصور درستی نیست. واژه Free وقتی در مورد نرم‌افزار به کار گرفته می‌شود به مفهوم آزاد بودن است – آزاد بودن در نسخه‌برداری، تغییر دادن و توزیع نرم‌افزار. در اصل آزاد بودن در این زمینه‌ها مستلزم داشتن دسترسی یکسان به کد منبع است. واژه Free Software ساخته ریچارد استالمن است. او نرم‌افزارهای مهمی برای یونیکس طراحی کرد و آنها را تحت مجوز GPL خود در اختیار عموم قرار داد. سپس "بنیاد نرم‌افزار آزاد" یا FSF را برای ترویج این برداشت از آزادی، ایجاد کرد. طبق مدلی که توسط FSF ایجاد شد، لینوز توروالدز، لینوکس را تحت مجوز نرم‌افزار آزاد GPL توزیع کرد.

دیگر پیشگامان حرکت نرم‌افزار آزاد مثل "اریک ریموند" و "لاری آگوستین" نسبت به برداشت اشتباه از معنی کلمه Free در انگلیسی نگران بودند. چرا که آنها معتقد بودند معانی دیگر این کلمه (مثل رایگان) شرکتهای بزرگ نرم‌افزاری را بدون دلیل نسبت به این مدل توسعه نرم‌افزار به وحشت می‌اندازد. ریموند اقدام به نوشتن مقاله تاریخی خود با عنوان "کاتدرال و بازار" در شرح و توصیف طراحی نرم‌افزارهای با کیفیت توسط جوامعی از برنامه‌نویسان، نوشته است. او برای توصیف این نرم‌افزارها واژه "متن‌باز" را ابداع کرد، چرا که منطبق بر هیچ مدل اقتصادی نبود و هنوز هم دسترسی به کد منبع شرط لازم برای این فرآیند طراحی و توسعه محسوب می‌شود.

---

[1] Free Software

همچنین این دسته از پیشگامان به این نتیجه رسیدند که موفقیت متن‌باز وابسته به توصیفی روشن از آنچه که در مجوز متن‌باز آمده است، می‌باشد. بروس پرنز، مدیر پروژه انجمن دبیان (Debian) برای عقد یک قرارداد اجتماعی با انجمن نرم‌افزار آزاد تلاش فراوانی کرده بود (در مورد دبیان در فصل ۵ بطور مفصل صحبت خواهد شد فعلاً به همین اکتفا می‌کنیم که دبیان یکی از توزیعهای لینوکس است که توسط جامعه برنامه‌نویسان متن‌باز گسترش یافته است.) او و دیگر پیشگامان حرکت متن‌باز توافق کردند که این قرارداد به عنوان "تعریف رسمی متن‌باز" یا OSD برگزیده شود. به همین دلیل است بروس پرنز را بعنوان طراح OSD می‌شناسند.

قابل ذکر است که افرادی مثل ریچارد استالمن با تغییر نام به متن‌باز موافق نبودند. علت این امر را در قسمتهای بعدی این فصل، وقتی GPL و مجوزهای دیگر را تشریح می‌کنیم، بررسی خواهیم کرد.

تعریف متن‌باز

به زبان ساده مجوز متن‌باز حقوق هرکس را در هرجا با هر هدفی شامل استفاده، نسخه‌برداری، تغییر و توزیع نرم‌افزار محافظت می‌کند. در عمل این کار مستلزم دسترسی آزادانه به کد منبع است. این حقوق الزامی بطور رسمی در OSD تدوین شده‌اند و توسط OSI منتشر شده‌اند (Open Source Initiative یا OSI یک شرکت غیرانتفاعی است.) انجمن متن‌باز، OSD را به عنوان مرجعی برای تشخیص مجوزهای واقعی متن‌باز می‌شناسد. مجوزهایی که با مطابق OSD ارائه می‌شوند، ملزوماتی را که در بالا شرح داده شد، رعایت می‌کنند.

یک مسأله خیلی مهم که در اینجا مطرح می‌شود این است که OSD خودش یک مجوز نیست. نویسنده اولین نسخه OSD از آن اینگونه یاد می‌کند: ویژگیهایی که یک مجوز واقعی باید داشته باشد تا نرم‌افزارهایی که تحت این مجوز توزیع می‌شوند، بتوانند از نظر قانونی متن‌باز باشند. جزئیات بیشتری از OSD را می‌توانید در قسمت مرجع در انتهای کتاب بیابید.

OSI یک نشان اعتباری بعنوان شاخص "متن‌باز قانونی" طراحی کرده است. وقتی روی یک بسته نرم‌افزاری "OSI Certified" را می‌بینید، می‌توانید مطمئن شوید که مجوزی که آن نرم‌افزار تحت آن توزیع شده است، با اصول OSD سازگار است.

چنین تعریف استانداردی برای مجوزهای اختصاصی وجود ندارد. هر شرکتی مجوز مربوط به خود را طراحی و استفاده می‌کند. در موقع کار با هر یک از این گونه مجوزها، لازم است یک حقوقدان شرایط آن را با دقت مطالعه کند و همواره سازمان را در زمینه رعایت آنها راهنمایی کند.

در سالهای قبل مردم راجع به مفهوم و معنای متن‌باز سئوالات متعددی از من می‌پرسیدند. من به این نتیجه رسیدم که بهترین جوابی که می‌توان به آنها داد خلاصه‌ای از تعریف مطرح شده در OSD است . در زیر خلاصه‌ای از آن آمده است. اگر تعریف جامع‌تری می‌خواهید، به وب سایت OSI (www.opensource.org) مراجعه کنید. همه این شرایط باید در یک مجوز لحاظ شده باشند تا بتوان آن را متن‌باز تلقی کرد و نرم‌افزاری که تحت آن مجوزدهی می‌شود، تأیید شده OSI باشد. شرایط OSI عبارتند از:

۱. **توزیع مجدد رایگان** - مجوز متن‌باز نمی‌تواند صاحب جواز را از توزیع مجدد نرم‌افزار بطور رایگان منع کند. صاحب جواز مجاز است که تعیین کند آیا نسخه‌برداری هزینه دربردارد یا خیر و مقدار این هزینه چقدر خواهد بود. معمولاً فراهم‌کنندگان نرم‌افزار بابت رسانه، کتاب راهنما و خدمات پشتیبانی محصول نرم‌افزاری متن‌باز هزینه دریافت می‌کنند. در عمل این شرط منجر به عرضه نرم‌افزار بصورت رایگان و یا با هزینه کم می‌شود.

۲. **کد منبع** - کد منبع برنامه باید موجود و در دسترس باشد. در اکثر موارد کد منبع همراه با قسمت باینری (برنامه اجرایی) ارائه می‌شود، ولی همیشه این گونه نیست. به هرحال اگر کد منبع همراه با باینری ارائه نشود، باید راهی برای دریافت آن بصورت رایگان موجود باشد. در این حالت چون کد منبع موجود است، هر فردی یا به عبارت صحیح‌تر هر فردی دارای مهارتهای لازم می‌تواند خطاها را کشف کند یا قابلیتهای نرم‌افزار را افزایش دهد.

۳. **کارهای اقتباسی** - مجوز باید به نرم‌افزار اجازه تغییر و به نرم‌افزار تغییر یافته اجازه پخش مجدد بدهد. اگر نرم‌افزار متن‌بازی را در اختیار بگیرید، در تغییر دادن آن آزادید و همچنین می‌توانید نسخه تغییر یافته را مجدداً توزیع کنید.

۴. **سالم بودن کد منبع نویسنده برنامه** - در روند توسعه در جامعه متن‌باز، درختواره رسمی کدهای منبع (مجموعه کدهای منبع تولید شده و اصلاح شده مورد تایید) یک پروژه، توسط نگهدارنده آن کنترل می‌شود. در مواردی ممکن است نگهدارنده با اعمال برخی تغییرات به درختواره رسمی کدهای منبع موافق نباشد و آنها را رد کند، ولی برنامه‌نویس خواهان توزیع آن تغییرات است. از نقطه نظر تشخیص و مجوزدهی لازم است که راه مشخصی برای جدا کردن کدهای منبع رسمی از گونه‌های تغییر یافته وجود داشته باشد. شرط جامعیت[1] در OSD تضمین می‌کند که شما می‌دانید چه کسی مسئول کد منبعی است که از آن استفاده می‌کنید. بنابراین نویسنده اصلی فقط در زمینه کدی که خودش نوشته است تحسین یا سرزنش می‌شود. برخی از مجوزهای متن‌باز ملزم می‌کنند که تغییرات از طریق وصله‌ها اعمال شوند. در این حالت کد اصلی بدون تغییر، توزیع می‌شود و تغییرات در فایلهای جداگانه‌ای ارائه می‌شوند. این دو در زمان کامپایل ترکیب می‌شوند تا نرم‌افزاری با قابلیتهای جدید ایجاد شود.

۵. **نبود تبعیض در برابر اشخاص یا گروه‌ها** - مجوز متن‌باز نمی‌تواند استفاده یا توزیع مجدد نرم‌افزار و کد منبع آن را برای هیچ شخص یا گروهی منع کند. این شرط هیچ محدودیتی ندارد، حتی اگر از برخی کاربران نرم‌افزار متنفر هم باشیم (مثلاً نمی‌توانید تروریستها را از استفاده از نرم‌افزار منع کنید). از نقطه نظر تجاری، این شرط این معنی را دربردارد که نمی‌توانید یک رقیب را از استفاده و توزیع مجدد نرم‌افزارتان منع کنید.

۶. **نبود تبعیض در کاربرد** - این شرط مانع کنترل یا محدودکردن نحوه استفاده از نرم‌افزار است. صادر کننده مجوز نرم‌افزار نمی‌تواند عقاید فرهنگی، اجتماعی و یا سیاسی خود را روی مجوزهای نرم‌افزار تحمیل کند. مجوز نمی‌تواند مانع تجاری کردن نرم‌افزار شود.

---

[1] integrity

۷. **توزیع مجوز** - در مدل متن‌باز، تنها مجوزی که می‌توان بر نرم‌افزار تحمیل کرد، مجوز متن‌باز است. لزومی به امضاء موافقتنامه عدم افشای مدارک[1] یا صدور مجوز ثانویه برای بدست آوردن نرم‌افزار نیست.

۸. **مجوز نباید خاص یک محصول باشد** – نرم‌افزارهای متن‌باز نمی‌تواند محدود به استفاده به همراه محصولات خاص شود. مثلاً نرم‌افزارهای متن‌باز محدود به استفاده با لینوکس نیستند.

۹. **مجوز نباید با دیگر نرم‌افزارها درگیر شود** - برای اینکه مجوزی با OSD سازگار باشد، نباید شرایط خود را بر نرم‌افزارهای دیگر توزیع‌شده به همراه آن تحمیل کند. این امکان، شرکتها را به توزیع نرم‌افزارهای متن‌باز و اختصاصی روی یک رسانه (مثلا لوح فشرده)، تشویق می‌کند.

این ۹ شرط برای همه مجوزهای متن‌باز ضروری هستند.

مالکیت معنوی و دوطرفه بودن

بعضی از مجوزهای متن‌باز بار مسئولیت دوطرفه بودن را بر صاحب جواز تحمیل می‌کنند. خیلی مهم است که این مفهوم و تأثیرات آن روی مالکیت معنوی به درستی درک شود. ولی اول باید بفهمیم که دوطرفه بودن[2]چیست و چرا تعداد زیادی از طرفداران نرم‌افزار آزاد اصرار دارند در مجوزهایشان از این مفهوم بهره گیرند. دوطرفه بودن شما را ملزم می‌کند که تمامی کارهای مشتق شده از کار اصلی را تحت مجوز متن‌باز مشابهی در اختیار نویسنده اصلی کار و سایر کاربران قرار دهید. مجوزی که لینوکس تحت آن ارائه شده است، GPL نام دارد که شرط دوطرفه بودن را در بردارد.

FSF از واژه copyleft برای توصیف دوطرفه بودن در GPL استفاده کرده است. علت این انتخاب نشان دادن رابطه آن واژه با قانون حق انحصاری کپی[3] صورت گرفته است. copyleft ، روشی عمومی برای تبدیل یک برنامه به نرم‌افزار آزاد و همچنین ملزم کردن برنامه‌نویسان به توزیع

---

[1] Non Disclosure Agreement

[2] reciprocity

[3] copyright

نسخه‌های تغییر یافته آن می‌باشد. برای این کار ابتدا نرم‌افزار [1]می‌شود، سپس تحت یک مجوز توزیع گردیده تغییراتی در آن ایجاد می‌شود و مجدداً کد برنامه تغییر یافته یا هر برنامه دیگری که از آن مشتق می‌شود، توزیع می‌گردد. بنابراین کد منبع و آزادیهای مربوط به آن قانوناً جدانشدنی می‌شوند.

دوطرفه بودن (copyleft) به طراح یک نرم‌افزار، امکان بهره‌گیری از تلاشهای شما را می‌دهد، همان‌طور که شما از مزایای در اختیار داشتن کد منبع وی بهره می‌برید. ارائه‌دهندگان نرم‌افزار آزاد، انگیزه زیادی برای ارائه محصولات خود تحت این مدل دارند، زیرا می‌دانند که نرم‌افزار آزاد بهتری بدست می‌آورند.

شرط copyleft شامل استفاده نرم‌افزار نشده و فقط زمانی معنی پیدا می‌کنند که نرم‌افزار تغییر یافته مجدداً توزیع می‌شود. بنابراین اگر شما روی کامپیوترهای فروخته شده به مشتریان خود از لینوکس به عنوان سیستم‌عامل استفاده می‌کنید، الزامات دوطرفه بودن شامل حال شما نخواهد شد و زمانی اعمال می‌شوند که در زمان استفاده از یک کار مشتق شده، برنامه اصلی را تغییر داده باشید - به همان تفسیری که در قانون حق کپی آمده است. بنابراین اگر صرفاً لینوکس را همان طور که دریافت کردید بین مشتریان توزیع کنید، این الزامات موجود در مجوز لینوکس شامل حال شما نمی‌شوند.

اگر لینوکس را تغییر دهید و نسخه تغییر یافته را توزیع کنید، باید آن تغییرات را تحت قوانین GPL مجوزدهی کنید. این تنها حالتی است که الزامات دوطرفه بودن به شما تعلق خواهد گرفت. ولی این موقعیت خود یک فرصت طلایی است.

وقتی نرم‌افزاری را که از برنامه‌نویسی دریافت کرده‌اید، تغییر می‌دهید، بهتر است در مورد اینکه آیا این عمل مشمول قاعده کارهای مشتق شده می‌شود یا با یک حقوقدان متخصص امور مالکیت معنوی مشورت کنید. طبق تمامی مجوزهای متن‌باز دوطرفه، شما باید کار جدید خود را که با اعمال تغییرات در نرم‌افزار دریافتی ایجاد شده است، تحت همان مجوز متن‌باز (یا مجوزی مشابه) در اختیار سایرین بگذارید.

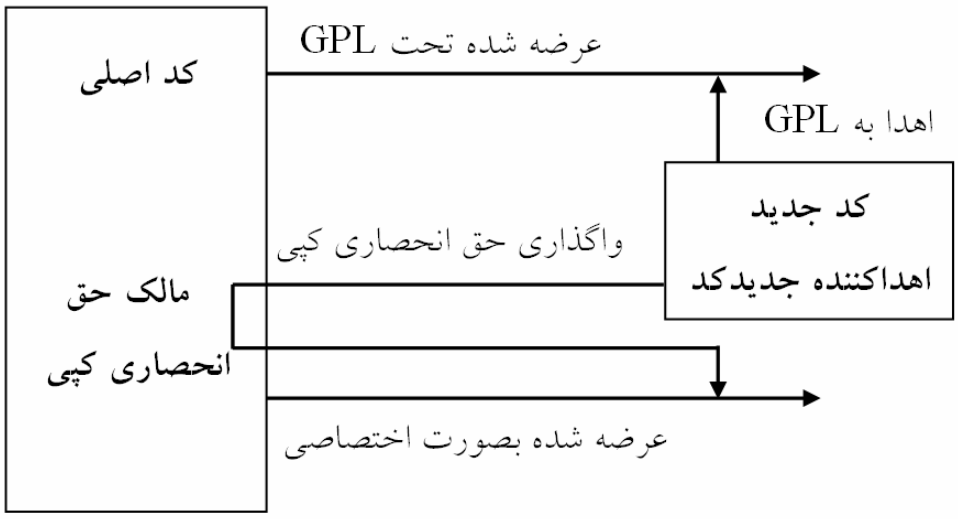
---

[1] copyleft

مالکیت حق انحصاری کپی و مجوز دوگانه

هر دارنده حق انحصاری کپی می‌تواند نرم‌افزار انحصاری خود را تحت یک یا چندین مجوز، مجوزدهی کند. بعد از آن هر صاحب جواز (خریدار) می‌تواند انتخاب کند که نرم‌افزار را تحت چه مجوزی بپذیرد. در بعضی از نرم‌افزارها صاحب جواز به شما امکان انتخاب بین مجوزدهی نرم‌افزار تحت یک مجوز متن‌باز دوطرفه مثل GPL و یا مجوزهای تجاری که باید برای آنها حق‌الامتیاز پرداخت شود، را می‌دهد. چنین استراتژیهای بازاریابی مجوز دوگانه، در پاسخ به آن دسته از مشتریانی توسعه یافته است که تمایلی به پذیرش الزامات دوطرفه بودن را نداشته و از آن اجتناب می‌ورزند.

لینوکس تحت یک مجوز دوگانه ارائه نشده است، ولی تعدادی از نرم‌افزارهای متن‌باز مهم که تحت لینوکس اجرا می‌شوند،این گونه‌اند مثلاً "هانس رایزر"، فایل سیستم خود به نام ReiserFS را تحت چند مجوز ارائه داده است. این امر امکان به‌کارگیری ReiserFS توسط سیستم عامل لینوکس تحت قوانین GPL را می‌دهد. رایزر همچنان می‌تواند فایل سیستم خود را تحت یک مجوز اختصاصی نیز مجوزدهی کند. توسعه‌دهندگان سیستم‌عاملهای دیگر با رایزر برای گسترش فایل سیستم وی کار می‌کنند. تفاوت کار در این است که رایزر می‌تواند در مقابل فایل سیستم خود هزینه درخواست کند و توسعه‌دهندگان می‌توانند ساختار پشتیبانی مؤثرتری برای این فایل سیستم بوجود آورند. این مورد یک مثال از موارد بسیاری استفاده از مجوز دوگانه است. شکل ۳-۱ روند پیاده‌سازی یک مدل مجوز دوگانه را نشان می‌دهد.



شکل ۳-۱: واگذاری حق انحصاری کپی در مجوزدهی دوگانه

توسعه‌دهندگان متن‌باز برای اینکه هنگام مجوزدهی دستشان باز باشد، اغلب از ارایه‌دهندگان کد درخواست می‌کنند که حق انحصاری کپی کد را در اختیار آنها قرار دهند. با این روش توسعه-دهنده اصلی حق دارد در مورد استراتژی مجوزدهی برای نسخه توسعه‌یافته تصمیم بگیرد.

لینوس تورووالدز نیازی به دراختیار گرفتن حق انحصاری پیشرفتهای لینوکس را ندارد. ولی FSF آماده پذیرش حقوق انحصاری کدهای متن‌باز را دارد و به این ترتیب این بنیاد می‌تواند مدافع بخش عمده‌ای از کارهایی باشد که مجوز آنها را در اختیار دارد.

برخی ارائه‌دهندگان کد، از اختصاص دادن حق انحصاری خود به هر فرد دیگری خودداری می‌کنند. مدیریت و نگهداشتن حساب مالکیت حقوق انحصاری کدها، یکی از دردسرهای عمده هر پروژه متن‌باز است.

مجوزها – متن‌باز و غیر متن‌باز

پس از درک OSD، دوطرفه بودن و چندین مورد مجوز دوگانه مرتبط با نرم‌افزار، اکنون می‌توانیم نگاهی بر چندین مجوز واقعی داشته باشیم. این قسمت با مجوزهای متن‌باز آغاز می‌شود و همان‌طور که پیش می‌رویم دو مجوز بسیار مهم GPL و LGPL ، به طور مبسوط، شرح داده می‌شوند.

مجوزهای متن‌باز

جدول ۳-۱ برخی از مشهورترین مجوزهایی را که مورد تأیید OSI هستند، شرح می‌دهد. شما و پرسنلتان شما می‌توانید با عضو شدن در سرویسهای خبری مربوطه، از جدیدترین مجوزهای مورد تأیید OSI مطلع شوید. جالب است بدانید که متن مجوزها توسط مجوز متن‌باز خاصی کنترل نمی‌شود و اگر متن مجوزی قابل تغییر باشد آنگاه لزومی ندارد با مجوز تطابق داشته باشد.

جدول ۳-۱ مجوزهای مشهور متن‌باز

مجوز	کپی لفت؟	توضیحات
مجوز نرم‌افزار آپاچی	خیر	این مجوز مورد استفاده سروروب آپاچی می‌باشد. این مجوز شبیه MIT و BSD است. تفاوت اصلی آن اینجاست که باید ذکر شود که نام تجاری آپاچی به همراه نرم‌افزار مجوزدهی نمی شود. یعنی اگر شما کد منبع آپاچی را تغییر دهید اجازه ندارید برنامه حاصل را "آپاچی" بنامید.
مجوز هنری	خیر	این مجوز برای زبان برنامه‌نویسی[1] طراحی شده است. این مجوز در کل مبهم است و موارد متناقض زیادی دارد. این تناقضها باعث شده است اغلب از آن پرهیز شود و بجای آن از مجوزهای GPL و MIT استفاده شود.
مجوز BSD	خیر	این مجوز از آنجایی که اجازه تغییر و توزیع برنامه بدون انتشار کد منبع را می‌دهد، بسیار طرفدار دارد. مجوز BSD اجازه می‌دهد که کد منبع را محرمانه نگه دارید و سپس آنرا تحت یک مجوز اختصاصی منتشر کنید. گرچه این مجوز بسیار جذاب به نظر می‌رسد، اما خواص تشویق‌کننده کار گروهی موجود در GPL را ندارد. در نسخه قدیمی این مجوز باید هربار که از این مجوز استفاده می‌شد، نام دانشگاه کالیفرنیا هم آورده می‌شد. این بند تبلیغ کننده، باعث می‌شود با GPL سازگار نباشد که در نسخه‌های کنونی حذف شده است.
مجوز کلی عمومی گنو (GPL)	بلی	به بحث مربوط به GPL که در ادامه آمده است مراجعه کنید. این مجوز کرنل لینوکس است. اغلب پروژه‌های نرم‌افزاری از این مجوز استفاده می‌کنند.
مجوز کلی عمومی کوچکتر گنو (LGPL)	بلی	به بحث مربوط به LGPL که در ادامه آمده است مراجعه کنید. مشابه GPL است، اما اجازه می‌دهد نرم‌افزارهای غیرآزاد را به برنامه خود پیوند دهید. همچنان لازم است تغییرات انجام شده در صورت توزیع،

[1] perl

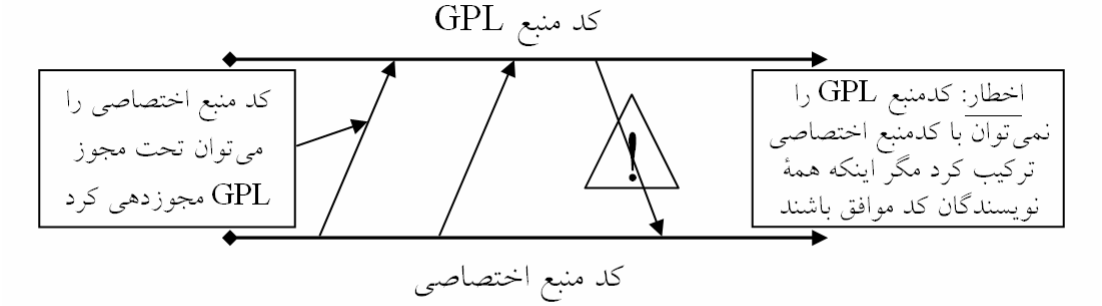
مجوز	کپی لفت؟	توضیحات
		منتشر شود.
مجوز عمومی IBM	بلی	یک مجوز تجاری کپی لفت مشابه مجوز عمومی موزیلا که تمهیدات بیشتری برای جبران خسارت درنظر گرفته است. اگر کسی نسبت به کد اصلی یا کدهای مشتق شده ادعایی مطرح کند، مسئولیت آن فقط به گردن مشتق کننده کد است و کسانی که در این بین بهبودهایی روی برنامه مشتق شده یا اصلی داده‌اند متهم نخواهند شد.
مجوز عمومی موزیلا (MPL)	بلی	موزیلا نسخه متن‌باز مرورگر وب Netscape (که اکنون جزو AOL شده است) می‌باشد. MPL از مشتقات NPL (مجوز عمومی Netscape) می‌باشد. برای اطلاعات بیشتر در زمینه NPL به جدول ۳-۲ مراجعه کنید. MPL توسط Netscape طراحی شد تا برخی از محدودیتهای NPL را برچیند. MPL اولین مجوز تجاری متن‌باز است و اکثر مجوزهای از این دست از این مجوز مشتق شده‌اند .
مجوز MIT	خیر	مجوزی است بسیار ساده که هیچ محدودیتی در استفاده از کد منبع ندارد. تنها لازمه آن، افزودن متن مجوز به تمامی نسخه‌ها می‌باشد. اگر قصد دارید مالکیت کد خود را حفظ کنید و در عین حال هرگونه استفاده از آن را نیز آزاد گذارید، این مجوز انتخاب خوبی است.
مجوز Python	خیر	Python یک زبان برنامه‌نویسی شیء گرای تفسیری می‌باشد که در بین جوامع لینوکس و یونیکس از محبوبیت خاصی برخوردار است. مجوز Python مخصوص Python است ولی بسیار آزاد است. کافیست نام مجوز Python را در کارهای مشتق شده خود ذکر کنید.
مجوز عمومی Qt (QPL)	بلی	Qt یکی از کتابخانه‌های مرکزی میزکار KDE است. عملکرد آن مشابه Motif است. این مجوز ملزم می‌کند که هر نرم‌افزاری که به آن پیوند می‌خورد کد منبع خود را در دسترس قرار دهد. QPL مشابه LGPL است با این تفاوت که دومی اجازه می‌دهد کدهای اختصاصی با کتابخانه‌ها پیوند برقرار کنند.
مجوز منبع استاندارد صنعتی SUN (SISSL)	خیر	شرکت Sun این مجوز را در پروژه OpenOffice برای بخشهایی که از قالب فایلی XML و توابع داخلی OpenOffice استفاده می‌کنند، اعمال کرده است. این مجوز به این منظور تعبیه شد تا این دو استاندارد همواره باز بمانند. این مجوز همچنین تضمین می‌کند که تمامی

مجوز	کپی لفت؟	توضیحات
		تغییرات اعمال شده به آنها در اختیار عموم قرار گیرد. از این گونه مجوزها زمانی استفاده می‌شود که در عین توسعه به روش متن‌باز نیاز به نوعی کنترل بر روی استانداردهایی که این توسعه در برمی‌گیرد احساس شود.

اگر چه این جدول مشهورترین مجوزهای متن‌باز را لیست کرده است، لازم است که نگاه جامع‌تری به مجوزهای اصلی copyleft مثل LGPL و GPL داشته باشیم.

LGPL و GPL

مشهورترین مجوز متن‌باز GPL یا GNU General Public License (به معنای مجوز عمومی کلی گنو) است. GPL توسط ریچارد استالمن به عنوان راهی برای جلوگیری از سوء استفاده از مفاهیم نرم‌افزار آزاد، تألیف شده است (برای مطالعه متن کامل این مجوز به ضمیمه ۳ مراجعه کنید.) وقتی نرم‌افزار آزاد و در اختیار عموم باشد، براحتی می‌توان آن را در اختیار گرفت و آن را به‌صورت اختصاصی (انحصاری) مورد استفاده قرار داد. استالمن که به این موضوع واقف بود قصد داشت کاری کند که "بده بستان" اینگونه نرم‌افزارها یا همان دوطرفه بودن ضمانت شود. اگر شما از نرم‌افزار متن‌بازی بهره می‌برید، تغییراتی یا اضافات شما به آن باید به سایرین نیز سود رساند.



شکل ۳-۲: ترکیب کد منبع GPL و اختصاصی

GPL تضمین می‌کند که زنجیره آزادی نرم‌افزار هرگز شکسته نخواهد شد. این مجوز ملزم می‌کند که هرگونه تغییری که در کد منبع GPL ایجاد می‌کنید و آنچه توزیع می‌کنید باید در اختیار جامعه متن‌باز قرار گیرد. GPL اولین مجوز copyleft بود. شکل ۳-۲ دو نمونه از "درختواره کد منبع" را نشان می‌دهد: یکی GPL و دیگری اختصاصی. این شکل همچنین ملاحظات مربوط به روند تغییر کد منبع را نیز نشان می‌دهد.

متاسفانه بعضی افراد به اشتباه تصور می‌کنند که منظور از GPL این است که هرچیزی که متأثر از اجرای کد GPL است نیز باید GPL باشد. این تصور کاملا غلط است و بهترین مثال از این حالت سیستم‌عامل لینوکس است. کرنل لینوکس تحت GPL مجوزدهی شده است. امروزه برنامه‌های کاربردی بسیاری هستند (مانند اوراکل) که تحت لینوکس اجرا می‌شوند، ولی GPL نیستند. بخش سوم این کتاب، به جزئیات نوشتن نرم‌افزارهایی که در ارتباط نزدیک با GPL هستند، ولی GPL نیستند، می‌پردازد.

مجوز دیگری که از خانواده خیلی نزدیک GPL است، LGPL یا Lesser GNU general Public License (به معنای GPL کوچک) است. LGPL برای حل مشکلات استفاده از کتابخانه‌هایی طراحی شد که تحت GPL مجوزدهی شده بودند. GPL هر برنامه تجاری را که از کتابخانه‌های مشترک[1] استفاده می‌کند، ملزم به داشتن مجوز GPL می‌کند. ولی LGPL چنین شرطی نمی‌گذارد. مثلا در محیط لینوکس برنامه‌های کاربردی زیادی هستند که وابسته‌اند به کتابخانه اصلی زمان اجرای سیستم، یعنی glibc با مجوزدهی glibc تحت LGPL، برنامه‌های کاربردی تجاری می‌توانند بدون نگرانی از اینکه مجبور به مجوزدهی تحت GPL شوند، با این کتابخانه پیوند برقرار کنند و از امکانات آن بهره‌مند شوند. وقتی که قصد دارید فناوری طراحی کنید که متن‌باز باشد و درعین حال می‌خواهید نرم‌افزارهای تجاری نیز قادر به استفاده از آن فناوری باشند، LGPL راه‌حل خوبی است. ولی دقت داشته باشید هنگام تغییر دادن کد منبع LGPL قوانین معمول GPL حاکم هستند؛ تغییرات و پیشرفتها در خود کتابخانه، هنگام توزیع، باید تحت LGPL مجوزدهی شوند. از آنجاییکه LGPL اغلب به کتابخانه‌ها اعمال می‌شود، از آن گاهی به library GNU general public license نیز یاد می‌شود. شکل ۳-۳ نحوه اتصال یک نرم‌افزار اختصاصی به نرم‌افزار LGPL و چگونگی ترکیب آن دو برای بوجود آوردن یک برنامه کاربردی کامل را نشان می‌دهد.

[1] shared

شکل ۳-۳: برقراری پیوند بین نرم‌افزار اختصاصی و کد LGPL

اگرچه امروزه از LGPL در طیف وسیعی از نرم‌افزارها استفاده می‌شود، ولی باعث ایجاد سردرگمیهایی نیز شده است. به همین دلیل ریچارد استالمن اکنون علیه استفاده از آن و در عوض افزودن یک تبصره به GPL را توصیه می‌کند:

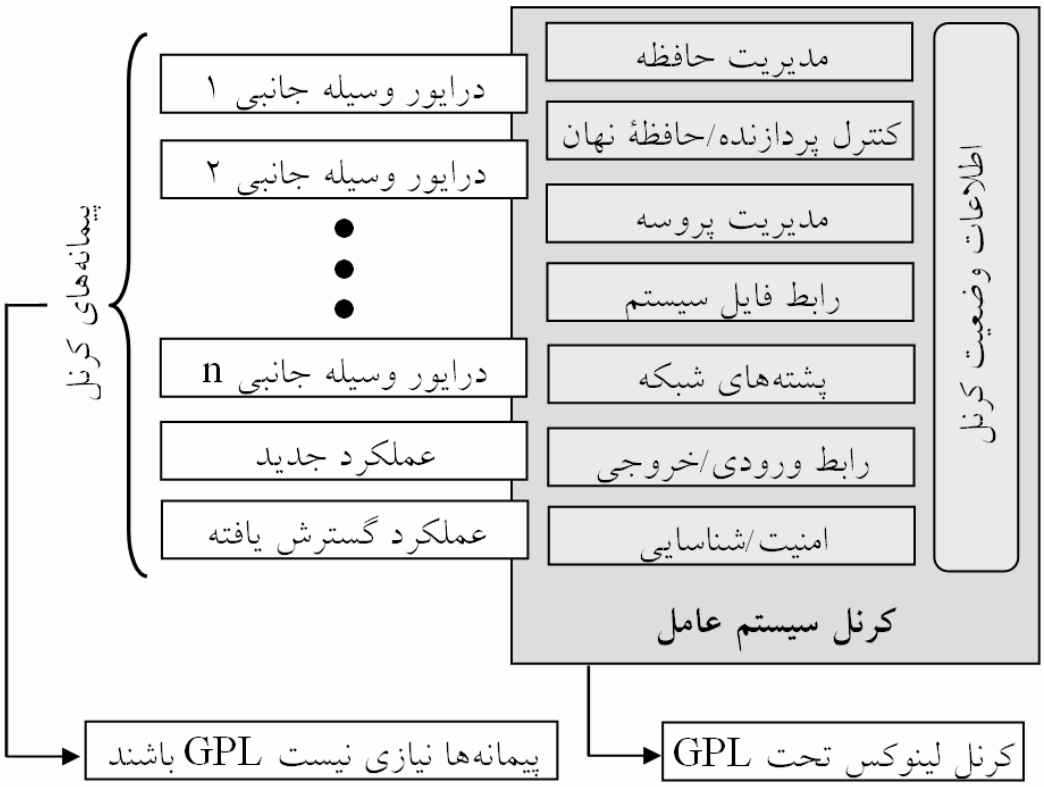
"به عنوان یک استثناء خاص، اگر کتابخانه‌ای را با دیگر فایلها جهت تولید یک برنامه قابل اجرا لینک کنید، این کتابخانه به خودی خود باعث نمی‌شود که برنامه اجرایی حاصل تحت پوشش GPL قرار گیرد. به هر حال، این استثناء موجب نقض دیگر دلایل اینکه چرا فایل اجرایی ممکن است تحت پوشش GPL قرار گیرد، نخواهد شد."

گرچه GPL و LGPL احتمالاً شناخته‌شده‌ترین و پرکاربردترین مجوزهای متن‌باز هستند، مجوزهای بسیار دیگری از این دست وجود دارند. بعضی از این مجوزها قبل از به صحنه آمدن OSD وجود داشتند. برخی از آنها طوری طراحی شده‌اند که محدودیتهای کمتری نسبت به GPL داشته باشند. از آنجایی که تقریبا بیش از ۳۰ مجوز ثبت شده وجود دارد و طراحی موارد جدید عموماً فقط گیج کننده خواهد بود. بنابراین بهتر است تا جایی که ممکن است از مجوزهای موجود استفاده کنید و از طراحی موارد جدید خودداری کنید.

GPL و پیمانه¬های کرنل لینوکس

همان طور که در فصل ۲ بیان شد، یکی از نقاط قوت نسخه ۲ کرنل لینوکس پشتیبانی از .ی کرنل بود. پیمانه¬های کرنل باعث می‌شوند که بتوان کرنل را که به مؤلفه‌های تابعی تقسیم کرد. بجای اینکه مجبور به کامپایل کردن دستگاه گردانها و عملکردهای جدید، بطور مستقیم در کرنل باشیم، پیمانه¬ها را می‌توان به محض نیاز، در زمان اجرا به کرنل "پیچ" کرد. وقتی یک ماژول در کرنل

بارگذاری (load) می‌شود، بخشی از کرنل می‌شود. شکل ۳-۴ مشابه یکی از تصویرهای فصل ۲ است، با این تفاوت که ماهیت ذاتی متن‌باز بودن پیمانه¬های کرنل را نیز مد نظر دارد.



شکل ۳-۴: پیمانه‌های کرنل لینوکس و GPL

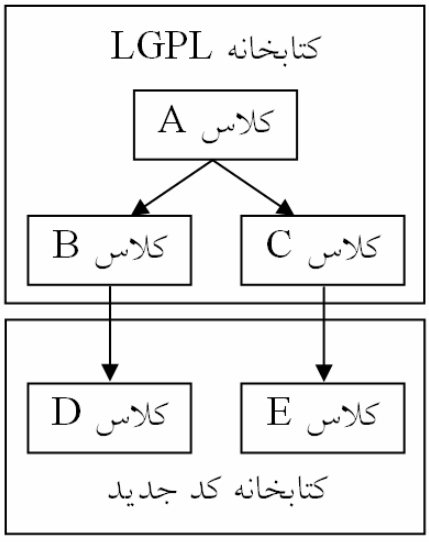
از نظر مجوز GPL، در اینجا یک مشکل پیش می‌آید. آیا GPL پیمانه¬ها را هم محدود می‌کند؟ از یک طرف ماژول بخشی از کرنل است و باید طبق قواعد GPL محدود شود و از طرف دیگر می‌توان گفت ماژول کد مجزایی است و از قوانین GPL مستثنی است. این ابهام تبدیل به یکی از بحثهای داغ در بین جامعه متن‌باز شده است. لینوس تورووالدز این مسأله را به طریقی روشن کرده است. او معتقد است که درصورتیکه پیمانه¬های کرنل، خود را به واسطهای استاندارد محدود کنند، مشمول محدودیت‌های GPL کرنل لینوکس نمی‌شوند. پیمانه¬های کامپایل شده نمی‌توانند فراخوانهای سیستم (توابعی که برنامه‌نویسان برای ارتباط با کرنل از آنها استفاده می‌کنند) جدیدی تولید کنند یا قسمتهای موجود کرنل را بازنویسی کنند. اگر چه به نظر می‌رسد این استدلال بتواند راه گریزی از GPL باشد، ولی به‌کارگیری آن عاقلانه به نظر نمی‌رسد. چرا که تورووالدز تنها مالک حق انحصاری کرنل نیست. این موضوع تا حدی پذیرفته شده است که اگر توسعه‌دهندگان کرنل اقدام به توسعه کارآیی کرنل از طریق به‌کارگیری استدلال "راه گریز پیمانه¬های کرنل" کنند، این راه گریز بسته

خواهد شد مثلا بواسطه یک تجدید نظر ساده در GPL. بنابراین این راه گریز را برای تغییر کرنل لینوکس بکار نبرید.

LGPL و وراثت کلاس

LGPL مجوزی است که اغلب درمورد کتابخانه‌ها استفاده می‌شود. این مجوز به کدهای اختصاصی امکان استفاده از کاراییهای کتابخانه‌ها را بدون ملزم بودن به رعایت قوانین GPL می‌دهد. مشکل دیگری در ارتباط با توسعه کلاس کتابخانه‌ها وجود دارد. همان طور که در شکل ۳-۵ قابل مشاهده است، زبانهای شیءگرایی مثل C++ بر پایه سلسه مراتبی از کلاسها ساخته شده‌اند. هر توسعه‌ای که روی سلسه مراتب صورت می‌گیرد، کارآیی جدیدی به کلاس پایه اضافه می‌کند. گرچه قوانین LGPL موقع استفاده از کاراییهای یک کتابخانه کاملاً واضح هستند، این قوانین زمان توسعه کارآیی یک کتابخانه محتوی کلاس [1] خیلی روشن و واضح نیستند.

در زمان نوشتن این کتاب بحث روی این موضوع در حال انجام است. از آنجایی که تاکنون جواب واضحی پیدا نشده است، هنگام توسعه کارایی یک Class Library موجود نباید LGPL به کار گرفته شود. البته این برای وقتی است که کد شما اختصاصی است.



شکل ۳-۵: LGPL و ارث‌بری کلاسها

[1] Class Library

مجوزهای غیر متن‌باز

تعداد اندکی مجوز غیرمتن‌باز یا مجوزهایی که منطبق باشرایط OSD نیستند، ولی بسیاری از ملزومات آن را ارضاء می‌کنند، وجود دارد. به خاطر داشته باشید که صرف اینکه کد منبع در دسترس شما است به این معنا نیست که مجوز آن شرایط OSD را برآورده می‌کند. این مجوزهای غیر متن‌باز عموماً برای حل موارد خاص تجاری طراحی شده‌اند. در موارد بسیاری ممکن است شرکت حق مجوزدهی کد را نداشته باشد، ولی بخواهد از توسعه اشتراکی و همگانی بهره ببرد. در موارد دیگری هم ممکن است در دسترس گذاشتن کد منبع به منظور کمک گرفتن از توسعه-دهندگان در اشکال زدایی برنامه‌های کاربردی صورت گیرد. در اینجا امکان لیست کردن فهرست جامعی از مجوزهای غیر متن‌باز وجود ندارد. اکثر مجوزهایی که امکان دسترسی به کد منبع را فراهم می‌کنند ولی OSD را ارضاء نمی‌کنند، در دسته‌بندی زیر قرار می‌گیرند:

۱) مالک می‌خواهد امکان اشکال‌زدایی یک برنامه کاربردی متکی به کتابخانه را بدهد. مثلاً مایکروسافت کد منبع را به اکثر کلاسهای پایه خود (MFC) توزیع می‌کند. این مسأله امکان ردیابی برنامه کاربردی را از طریق کد مایکروسافت به برنامه‌نویس می‌دهد. به هرحال، مایکروسافت تا این زمان صاحب یک مجوز متن‌باز واقعی نشده است.

۲) افراد یا شرکتها از حقوق کامل خود برخوردار نیستند. در موارد بسیاری اشخاص مایلند از مجوزهای متن‌باز استفاده کنند، ولی به واسطه قراردادهایی که با اشخاص ثالث بسته‌اند، محدود شده‌اند و نمی‌توانند چنین کاری انجام دهند. در چنین مواردی اغلب مجوز جدیدی خاص شرکت و یا آن محصول طراحی می‌شود.

۳) افراد یا شرکتها تمایل دارند بدون در اختیار گذاشتن مالکیتهای معنوی[1] خود، از یک انجمن برنامه‌نویسان بازخورد دریافت کنند. حالتهایی وجود دارد که شرکت می‌خواهد دسترسی به کد منبع وجود داشته باشد تا از مزایای بازخوردها و نظرات انجمنهای

[1] Intellectual Propertey

توسعه‌دهنده بهره گیرد. هرگونه تغییرات و حقوق نرم‌افزار باید به شرکت برگردد. مجوز "کد منبع اشتراکی" مایکروسافت در این رده قرار می‌گیرد. این گونه مجوزها مشکل بزرگی در جامعه توسعه متن‌باز بوجود آورده‌اند. به هرحال ممکن است برای شما مناسب باشند و این در صورتی است که مدل تجاری شما کاملاً وابسته به درآمد ناشی از مجوز نرم‌افزارها باشد و استفاده از مدل تجاری دیگری برای شما مقدور نباشد.

جدول ۳-۲ حاوی لیستی است از مجوزهای غیر متن‌باز مهمی که باید با آنها آشنا باشید. این مجوزها ملزومات OSD را برآورده نمی‌کنند.

جدول ۳-۲ مجوزهای مهم غیر متن‌باز

نام مجوز	توضیحات
کد منبع اشتراکی مایکروسافت	این مجوز اجازه دسترسی به کد منبع را می‌دهد، ولی شما را ملزم به بازگرداندن تغییرات به مایکروسافت می‌کند. تمامی حقوق این تغییرات متعلق به مایکروسافت خواهد بود. در این مجوز اجازه توزیع کد وجود ندارد.
مجوز عمومی Netscape (NPL)	NPL مختص Netscape است. Netscape این مجوز را برای خاتمه دادن به کشمکشهای مابین آن شرکت و شرکتهای ثالث ساخت. تا وقتی که موزیلا کد متن‌باز واقعی این مرورگر را فراهم کرد، NPL راهی پیش روی Netscape قرار داد تا بتواند کد را عرضه کند و از پشتیبانی جامعه برنامه‌نویسان بهره گیرد و در عین حال لطمه‌ای به قراردادهایش با شرکتهای ثالث وارد نگردد.
مجوز منبع جامعه Sun	این مجوز هم نوعی مجوز کد منبع اشتراکی است. می‌توان برای بهبود یک پروژه یا تکنولوژی برای آن کد نوشت و در اختیار آن قرار داد، اما همه کارهای عرضه شده باید از استانداردهای Sun تبعیت کنند. توزیع مجدد آن مستلزم پرداخت هزینه به شرکت SUN Microsystems است. همان گونه که در جدول قبلی اشاره شد، SUN مجوز دیگری نیز دارد که در قالب OSD می‌گنجد.

این مجوزها آن قدر مهم هستند که باید آنها را مدنظر داشته باشید. به خاطر داشته باشید که مجوزهای جدید مرتباً ساخته می‌شوند و باید دقت کنید و به روشنی بدانید که یک مجوز یا همه شرایط OSD را برآورده می‌کند یا یک مجوز غیر متن‌باز است (مورد تایید متن‌باز نمی‌باشد.)

صادر کردن و رمزنگاری

نرم‌افزاری که متن‌باز است، باید از قوانین مربوط به کنترل صادرات نیز پیروی کند. این مسأله خصوصاً برای نرم‌افزارهایی که قابلیتهای رمزنگاری دارند از اهمیت بیشتری برخوردار است، چرا که ایالات متحده آمریکا و چندین کشور دیگر روی صادرات اینگونه نرم‌افزارها محدودیتهایی اعمال کرده‌اند.

از آنجایی که نرم‌افزارهای متن‌باز در سراسر جهان به سادگی نسخه‌برداری و توزیع می‌شوند، هیچ مجوز متن‌بازی نمی‌تواند صریحاً از قوانین ایالات متحده تبعیت کند و آن را قید کند. چنین مجوزی با شرط پنجم OSD که بر عدم تبعیض بین افراد و گروه‌ها تاکید دارد، در تناقض است. یک مجوز متن‌باز نمی‌تواند عذری در برابر قانون بیاورد. هر کسی که نرم‌افزاری را می‌نویسد و آن را توزیع می‌کند، با توجه به اینکه از قوانین کنترل صادرات آگاه است، در برابر آن مسئول است و باید از آن قوانین پیروی کند.

این مسأله توزیع‌های متن‌باز لینوکس را با مشکل روبرو کرد. اوایل تنها راهِ ارائه کامل قابلیتهای رمزنگاری در لینوکس، انجام آن در خارج از ایالات متحده بود. برای مدتی نرم‌افزارهای خاصی در این زمینه (مانند SSL و سرورهای وب خاص) باید از طریق منابع و سایتهای خارج از ایالات متحده توسعه می‌یافتند و دریافت می‌شدند.

تغییرات اخیر در قوانین صادرات ایالات متحده به نرم‌افزارهای متن‌بازی که شامل کدهای رمزنگاری می‌شوند، امکان صادر شدن را می‌دهد. اکنون، تمام آنچه که لازم است مورد توجه قرار گیرد و انجام شود، اطلاع و تذکر ساده‌ای است به نمایندگیهای دولتها در این زمینه. با این روش، نرم‌افزارهای

رمزنگاری می‌توانند متن‌باز باشند و در عین حال از محدودیتهای قوانین نیز در امان بمانند. توضیحات بیشتری در این زمینه را می‌توانید در وب سایت www.bis.doc.gov ببینید.

از آنجایی که همیشه احتمال تغییر در محدودیتها و قوانین وجود دارد، خیلی از نرم‌افزارهای رمزنگاری خارج از ایالات متحده توسعه داده و نگهداری می‌شوند. بنابراین هر تغییری در قوانین صادرات این کشور تاثیر حفیفی بر لینوکس و برنامه‌های کاربردی متن‌باز دارد.

متدلوژی توسعه متن‌باز

اگر چه بحث مدلهای مجوزدهی معمولاً وقتی به متن‌باز می‌رسد، بسیار جالب توجه می‌شود، اما چیزی که واقعا جالب است، متدلوژی توسعه متن‌باز است. زیبایی واقعی متن‌باز مجوزهای آن نیست، بلکه فرایند توسعه آن است.

توصیف کلی متدلوژی توسعه متن‌باز، تعداد زیادی از پروژه‌ها را پوشش می‌دهد. اما خصوصیات و ویژگیهای خاص آن بسته به اندازه و کاربرد پروژه متغیر است. مقاله (و کتاب) "اریک ریموند" جزئیات کار وی را شرح می‌دهد. ریموند متحیر بود که چگونه فرایندی که به نظر می‌رسد هر قانون شناخته شده‌ای از توسعه سنتی نرم‌افزار (مدل کاتدرال) را نقض می‌کند، هنوز جواب می‌دهد. مقاله ریموند مدل بازار را که پیشنهاد وی بود، شرح می‌دهد. در زیر عناصر کلیدی این متدلوژی بدون اعمال پروژه آورده شده است. شما باید واژگان، ابزارها، نقش آفرینان و فرایند آن را بشناسید.

مجوز

گرچه فرایند توسعه مستقیماً درگیر مسائل مربوط به مجوز نیست، این مجوز است که محدودیتها و مرزهایی که به واسطه آنها فرایند توسعه شکل می‌گیرد، تعیین می‌کند. این شرایط زیربنای توسعه همگانی و اشتراکی را تضمین می‌کنند. از آنجاییکه GPL و LGPL و دیگر مجوزهایی که ملزومات دوطرفه بودن را رعایت می‌کنند، بر این اصل استوارند که تغییرات و اصلاحات باید به انجمن برگشت داده شود، بنیان و شالوده توسعه همگانی تضمین می‌شود.

روش ها

روش توسعه متن‌باز را می‌توان روش توسعه لینوکس نیز نامید. لینوس توروالدز یک نمونه برای نشان دادن این موضوع است که به روش بازار کار می‌کند و خوب هم کار می‌کند. این روش به صورتهای زیر قابل تصور است:

۱. **شرکت**

در این روش شرکت رهبری یک پروژه را برعهده می‌گیرد. این فرآیند معمولاً به این ترتیب است که تعدادی توسعه‌دهنده را به پروژه اختصاص داده می‌شود و با سرمایه‌گذاری زیرساخت پروژه شکل می‌گیرد و برای نگهداری پروژه اقدام می‌شود. حتی در این روش هم، شرکت یک برنامه‌نویس را به عنوان نگهدارنده اصلی انتخاب می‌کند. HP، IBM، SUN و شرکتهای بزرگ دیگری پروژه‌های متن‌باز را با این روش هدایت و رهبری می‌کنند.

۲. **بنیاد**

بنیادها اغلب برای سرمایه‌گذاری روی پروژه‌های بزرگ ایجاد می‌شوند. شرکتها و افرادی که مایلند به موفقیت این پروژه کمک کنند، می‌توانند از طریق این سازمان غیرانتفاعی، پروژه را یاری کنند. مواردی وجود دارد که یک شرکت رهبری یک پروژه را برعهده گرفته است و در عین حال بنیادی هم برای آن بنا کرده است. این معمولاً به این خاطر است که شرکتی که رهبری پروژه را برعهده دارد، برای سرمایه‌گذاری روی آن از منابع مالی کافی برخوردار نیست. بنیاد گنوم (GNOME)، مثال خوبی از این روش توسعه است. شرکتهای زیادی در بنیاد گنوم سرمایه‌گذاری کرده‌اند. مثلاً Ximian، مؤسسه‌ای است که رهبری پروژه میزکار گنوم را تا نهایی کردن بر عهده گرفته است.

۳. **کمیته**

از روش کمیته نیز برای پروژه‌های بزرگ استفاده می‌شود. کمیته‌ها معمولاً میدانی برای تصمیم‌گیری فراهم می‌کنند. این روش، رویکرد تصمیم گیری جمعی را (در مقابل رویکرد

معمولی شامل یک نگهدارنده) ارائه می‌کند. کامپایلرهای GCC که اغلب در لینوکس استفاده می‌شوند به این روش مدیریت می‌شوند.

۴. **افراد**

پروژه‌های کوچک معمولاً توسط افراد اجرا می‌شود. در واقع هزاران نوع از این پروژه‌ها وجود دارد. اکثر این گونه پروژه‌ها روی سایت Source Forge قرار دارند و اخیراً Savannah هم که توسط پروژه گنو ارائه شده است، به مخزنهای نرم‌افزار آزاد اضافه شده است. این گونه پروژه‌ها اغلب کمتر از پنج برنامه‌نویس دارند. بیشتر این پروژه‌ها این گونه آغاز می‌شوند که یک هَکِر در پی یافتن راه‌حلی برای مشکل خود است و شروع به برنامه‌نویسی می‌کند تا نیاز خود را برآورده کند و آن را متن باز می‌کند.

۵. **کرنل لینوکس**

لینوس توروالدز از این روش خاص برای توسعه بزرگترین پروژه متن‌باز (کرنل لینوکس) استفاده کرد. توروالدز این مدل را در طی سالها تکمیل کرده است. این روش مبتنی بر سلسله مراتبی از اطمینان در انجمن توسعه کرنل است. در بخش ۳ این کتاب شرح داده خواهد شد که چگونه می‌توان مدل نظری فرآیند توسعه کرنل لینوکس را در یک مؤسسه بکار گرفت.

۶. **گنو**

این روش توسعه خیلی جا افتاده نیست و اغلب ترجیح داده می‌شود که از روشهای قبلی استفاده شود. روش گنو از متدلوژیهای توسعه قراردادی برای تولید نرم‌افزار آزاد استفاده می‌کند.

صفات

صفات، مجموعه‌ای از خصوصیات هستند که در اغلب پروژه‌های متن‌باز مشترکند. این مجموعه مشخصات پایه مهمی برای شکل دادن رفتار جامعه برنامه‌نویسان هستند. اغلب پروژه‌های متن‌باز صفات زیر را بطور مشترک دارند:

۱. **نگهدارنده**

اکثر پروژه‌ها توسط یک نگهدارنده هدایت می‌شوند. نگهدارنده فردی است که پروژه را بنیان می‌نهد یا کسی است که توسط نگهدارنده قبلی برای هدایت پروژه معین می‌شود. نگهدارنده کارهای ارائه شده را رد یا قبول می‌کند، وصله‌ها را به کار می‌گیرند، کاستی را کنترل و مدیریت می‌کنند و در صورت لزوم با دیگر پروژه‌ها تعامل برقرار می‌کند. وقتی یک نگهدارنده دیگر علاقه‌ای به ادامه هدایت پروژه نداشته باشد یا وقت این کار را نداشته باشد، کار خود را به نگهدارنده جدیدی می‌سپرد. نگهدارنده جدید از بین افرادی انتخاب می‌شود که در پروژه‌ها شرکت فعال دارند و داوطلب این سمت هستند. مواردی هم وجود دارد که یک مؤسس کمیته‌ای برای پروژه تأسیس کرده است. همچنین درمواردی که هدایت پروژه برعهده شرکت است، ممکن است گروهی از کارمندان شرکت برای نگهداری آن انتخاب شوند.

۲. **احترام**

در طول زمان، احترام به افراد در جامعه برنامه‌نویسان به صورت یک شاخص در آمده است. اگر توسعه‌دهنده‌ای کم کاری کند، از سوی دیگر اعضاء مورد تذکر قرار می‌گیرد. انتقاد و تحسین اعضاء در صورت لزوم اعمال می‌شود. افرادی که فعال هستند، از احترام خاصی در این جامعه برخوردار هستند. افرادی مثل توروالدز جایگاه خاصی در انجمن دارند. شرکت‌کنندگان در یک پروژه قبل از اینکه پذیرفته شوند به دقت مورد بررسی قرار می‌گیرند و کسانیکه نتوانند همکاری فعال داشته باشند رد می‌شوند.

۳. **افراد به عنوان نگهدارنده**

حتی اگر شرکتها پروژه‌های متن‌باز را مدیریت کنند، نگهدارنده اغلب یک فرد است. اگر

نگهدارنده شرکت خود را تغییر دهد، ممکن است همچنان به عنوان نگهدارنده رسمی کد باقی بماند. در بعضی موارد براساس تغییر اولویتها در شرکتها تغییراتی رخ می‌دهد. در چنین شرایطی نگهدارنده‌ها مشعل را به دست دیگری می‌سپارند. مردم توروالدز را به عنوان نگهدارنده کرنل لینوکس می‌شناسند نه Transmeta (شرکتی که او در آن کار می‌کند) را.

۴. ابزارها

همه توسعه‌دهندگان جامعه به زیرساخت کاملاً ساخت‌یافته‌ای همراه با ابزارهای توسعه رایج عادت کرده‌اند. مثلاً CVS ابزار مدیریت کاملاً پذیرفته شده‌ای است و Bugzilla اغلب برای پیگیری اشکالات به کار می‌رود. قابل ذکر است با اینکه CVS ابزار مدیریتی است که در طیف وسیعی از آن استفاده می‌شود، برای کرنل لینوکس به کار نمی‌رود. کرنل لینوکس مبتنی بر وصله‌ها است. SourceForge و اکنون Savannah برای نرم‌افزارهای آزاد و متن‌باز زیرساخت توسعه مناسبی ارائه می‌دهند.

۵. عرضه زود هنگام و مکرر

یکی از تفاوتهای اساسی بین روش توسعه سنتی کاتدرال و روش توسعه بازاری متن‌باز، عرضه کارهای درحال توسعه است که در روش بازار وجود دارد. این حلقه‌های عرضه سریع و مداوم به توسعه‌دهندگان امکان تشخیص نقاط ضعف را در کمترین زمان می‌دهد که بدین وسیله می‌توان آنها را در فرایند توسعه از بین برد. توسعه‌دهندگان به تناوب از نسخه‌های عرضه شده زودهنگام در کارهای روزانه خود استفاده می‌کنند تا عیوب آنها را پیدا کنند. این ویژگی دلیلی است برای این سؤال که چرا محصولات متن‌باز نسبت به محصولات سنتی خیلی سریعتر و با کیفیت خیلی بالاتر توسعه می‌یابند.

۶. عرضه پروژه بعنوان محصول پس از آماده شدن

همان طور که در فصل ۲ گفته شد، بسیاری از شرکتها، نرم‌افزار را در طول مدت زمانی محدود که بستگی به فاکتورهایی نظیر فشار بازار و رقبا دارد، طراحی می‌کنند که معمولاً نتایج خوبی

دربردارد. در فرآیند توسعه متن‌باز عموماً وقتی انجمن اقدام به ارائه محصول می‌کند که نگهدارنده تشخیص دهد محصول آماده است. خیلی از شرکتها وقتی با پروژه‌های متن‌باز روبرو می‌شوند از فقدان طرح کلی برای کار احساس ناراحتی می‌کنند. به هر حال کسانی که این شرط را می‌پذیرند، از مزایای نرم‌افزار دارای کیفیت بیشتر، وصله‌های کمتر و یک معماری بهتر برای سیستم، بهره‌مند می‌شوند.

ترکیب این صفات متن‌باز با روشهای مختلف توسعه، یک فرایند توسعه همگانی موثری را ایجاد می‌کند. جا انداختن این فرهنگ همکاری در شرکت، زمان و شهامت می‌طلبد ولی ارزش آن را دارد. مثلاً لینوکس و وب سرور آپاچی را در نظر بگیرید به عنوان دو پروژه بزرگ متن‌باز که چنان رشد یافته‌اند که بار مدیریت بخش عظیمی از اینترنت را بر عهده دارند.

جمع بندی

در این فصل با جزئیات متن‌باز آشنا شدید و نحوه ارتباط آن با نرم‌افزار آزاد را درک کردید. ما OSD را به طور مفصل شرح دادیم. همچنین به تعدادی از مجوزهایی که شرایط OSD را برآورده می‌کنند، اشاره کردیم. در نهایت به این موضوع پرداختیم که چگونه جامعه فرایند توسعه مبتنی بر روش بازار را اجرا می‌کند و بعد از آن ویژگیهای کلیدی پروژه‌ها در جامعه ذکر شد. در بخش سوم بحث خود را در مورد متن‌باز ادامه می‌دهیم و سعی خواهیم کرد برای سؤالات زیر جواب پیدا کنیم.

۱. ریسکهای تجاری مرتبط با GPL و LGPL چیست و چگونه می‌توان آنها را مدیریت کرد؟

۲. متن‌باز چگونه بر مدلهای تجاری موجود تاثیر می‌گذارد؟

۳. شرکتها چگونه می‌توانند از روش توسعه بازار بهره گیرند؟

۴. شرکتها چگونه می‌توانند روی پروژه‌های متن‌باز کار کنند؟

۵. چه فرآیندهای تجاری کلیدی متأثر از توسعه متن‌باز هستند؟

در فصل بعد تعدادی از انجمنها و سازمانها در جامعه بزرگ متن‌باز را بررسی می‌کنیم و همچنین نقشی را که هریک برعهده دارند مورد، بررسی قرار می‌دهیم.

فصل چهارم

انجمنها و شرکتها

متن‌باز مجموعه‌ای از انجمنها است. در فصل سوم برخی از روشهای مختلف اجرای پروژه‌ها از روش شرکتی تا بنیادی شرح داده شد. نوع دیگری از سازمانها وجود دارند که برای لینوکس و حرکت متن‌باز خدمات پشتیبانی ارائه می‌کنند. داشتن درک بهتری از تمام انجمنهای موجود و آشنایی با آنها در درک عظمت این جنبش مؤثر است. همچنین تعاملات پیچیده‌ای بین شرکتهای مختلف وجود دارد که شما تنها در صورتی آنها را درک خواهید کرد که نسبت به این جوامع شناخت داشته باشید.

این فصل طرح کلی از انجمنهای شاخص را ارائه می‌دهد. این بحث را با انجمن کرنل لینوکس شروع خواهیم کرد. سپس نگاهی به برخی مؤسسات اصلی متن‌باز خواهیم انداخت و در نهایت تعدادی از سازمانهای صنعتی را بررسی می‌کنیم. همراه با هر سازمانی، هم اینجا و هم در قسمت مرجع انتهای کتاب، آدرس سایت مربوط به آن نیز آورده شده است.

فهرست کردن لیست همهٔ گروه‌ها، انجمنها، مؤسسات و سازمانهای موجود در این فصل مقدور نیست، ولی به هر حال برای شروع کافی است. در مورد برخی از این سازمانها و انجمنها می‌توان یک فصل کامل یا حتی یک کتاب کامل نوشت. هدفی که دنبال می‌شود این است که یک دید کلی به شما داده شود، نه شناخت عمیق نسبت به همه گروه‌ها و پروژه‌ها. همچنین نگاهی اجمالی به برخی پروژه‌های کلیدی متن‌باز که برای صاحبان تجاری خود درآمدزا بوده‌اند، خواهیم انداخت. اگر در حال توسعه برنامه‌های کاربردی بازرگانی هستید، مواظب باشید شما نفر بعدی که با تخریب تدریجی بازار خود روبرو خواهد بود، نباشید.

احتیاجی به خواندن کامل این فصل نیست. می‌توانید این فصل را به طور اجمالی بررسی کنید و جزئیات انجمنی را که فکر می‌کنید به کار و تجارت شما مربوط است، با تفصیل بیشتری مطالعه کنید. همچنین می‌توانید نگاهی به انجمنهایی بیاندازید که با آنها آشنایی ندارید. بعدها هنگام نیاز به مراجعه به سایت یک انجمن خاص، به این فصل مراجعه کنید.

لینوکس

همان طور که در فصلهای اول و دوم فرا گرفتید، لینوکس درحقیقت یک کرنل است. انجمن کرنل شامل گروه‌هایی است که بر خود کرنل، امور مختص پردازنده‌ها تمرکز دارند و روی مواردی مانند فایل سیستمها و دستگاه گردانها کار می‌کنند.

کرنل

http://www.kernel.org

kernel.org مرکز دنیای کرنل است، گروهی از مهندسین که بر عمیق‌ترین و پیچیده‌ترین جنبه‌های توسعه نرم‌افزار تکیه کرده‌اند و با تمام توان کار می‌کنند. همانند انجمنهای دیگر، این انجمن بر پایه مکالمات پست الکترونیکی از طریق بازتاب دهنده‌ها (به این معنی که ایمیلی که به یک آدرس فرستاده می‌شود به طور خودکار به دیگر اعضاء نیز فرستاده می‌شود) عمل می‌کند. ترافیک ایمیل بسیار بالا است و تنها کسانی که عضو گروه توسعه کرنل هستند، می‌توانند به این انجمن متصل شوند.

پردازنده‌ها

X86(IA-32)

خانواده پردازنده‌های IA-32 انجمن مجزایی ندارد. زیرا در دل تمام انجمنهای توسعه کرنل وجود دارد و به عنوان مبنا درنظر گرفته می‌شود.

خانواده پردازنده‌های ایتانیوم(IPF یا IA-64)

http://www.linuxia64.org

IPF جانشین برای خانواده پردازنده‌های IA-32 شناخته می‌شود. این رویدادی یک شبه اتفاق نخواهد افتاد. در آینده‌ای نزدیک IA-32 از نظر قیمت/کارآیی بخصوص در زمینه سیستمهای

کاربری پیشگام خواهد بود. IPF در ابتدا در زمینه محاسبات سنگین پیشرو خواهد بود و در آینده وقتی که برنامه‌های کاربردی نیاز به محاسبه ۶۴ بیتی دارند، تبدیل به پردازنده برتر خواهد شد و اکوسیستمی مبتنی بر کالا حول معماری آن رشد خواهد کرد.

لینوکس تحت IPF پروژه‌ای است که در ابتدا به رهبری "دیوید موزبرگر[۱]" و "استفان ارانیان"[۲] در آزمایشگاه‌های HP آغاز به کار کرد. اینتل و شرکتهای دیگری که ملاحظات عدم افشای اطلاعات محرمانه اینتل را رعایت می‌کردند نیز در این فضا کار می‌کردند. کمی بعد از شروع پروژه گروهی از شرکتها شامل HP، Intel، SGI، IBM و چند شرکت دیگر کنسرسیومی با نام گروه تریلیان[۳] تشکیل دادند. تریلیان به شرکتها اجازه می‌داد بدون افشاء ویژگیهای Itanium به عموم، در زمینه پشتیبانی لینوکس تحت آن پردازنده کار کنند. وقتی اینتل، Itanium را گسترش داد، گروه تریلیان منحل شد و کد لینوکس تحت IPF بصورت متن‌باز در دسترس همگان قرار گرفت.

شبیه سازIPF برای خانواده پردازنده IA-32 موجود می‌باشد. یک نکتهٔ عجیب در مورد این شبیه-ساز این است که به برنامه‌نویسان این امکان را می‌دهد که از کامپیوترهای کیفی[۴] و رومیزی ارزان قیمت خود برای برنامه‌نویسی لینوکس تحتIPF استفاده کنند. اجزای دیگری نیز روی وب سایت رسمی لینوکسِIPF موجود است.

اکنون، دیوید به عنوان نگهدارنده هستهٔ لینوکسIPF ایفای نقش می‌کند. او همچنین مثال خوبی است از نگهدارنده به عنوان یک فرد (بجای شرکت.)

Alpha

http:// www.linuxalpha.org

پردازندهٔ آلفا از پردازنده‌های مشهور در تاریخ لینوکس است. در واقع آلفا برای مدت زمان طولانی پردازندهٔ برتر برای لینوکس در برنامه‌های کاربردی محاسباتی و با کارآیی بالا بود.

---

[1] David Mosberger

[2] Stephone Eranian

[3] Trillian

[4] laptop

برخی ارائه‌دهندگان توزیعها برای پیاده‌سازی آلفا پشتیبانی ارائه داده‌اند، به هر حال Compaq قبل از اینکه به مالکیت HP درآید، قریب‌الوقوع بودن پایان عمر آلفا را اعلام کرده بود و IPF را جایگزین آینده آن شناخته بود. عمده مالکیت معنوی آلفا متعلق به اینتل است.

PowerPC

http://www.linuxppc.org/
http://www.linuxppc.com/

IBM و Motorola با هم PowerPC را تولید کردند. انجمن PowerPC عمدتاً بر ارائه پشتیبانی لینوکس برای کامپیوترهای مکینتاش که ریزپردازنده PowerPC را به کار می‌گیرند، تمرکز دارد. همچنین برخی از سیستمهای IBM را پشتیبانی می‌کند. برخی توزیعها نیز نسخه‌هایی برای این معماری طراحی کرده‌اند.

PA- RISC

http://www.parisc-linux.org

PA-RISC خانوادهٔ پردازنده‌ای است که توسط شرکت HP توسعه یافته است. HP سرمایه‌گذاری عظیمی نیز در IPF کرده است و همراه با اینتل پیش می‌رود. این شرکت، در حال توسعهٔ PA-RISC برای IPF به عنوان ریزپردازندهٔ استراتژیک آینده است. گروه Puffin که یک شرکت کوچک برنامه‌نویسی است، انتقال هستهٔ لینوکس به پردازندهٔ PA-RISC را بنیان نهاده است. HP کمک خود را به شکل پشتیبانی توسعه و مستندسازی ارائه داد. گروه Puffin که به کمک LinuxCare شکل گرفت، تاکنون فعالیتهای (برنامه‌نویسی) زیادی برای لینوکس تحت PA-RISC انجام داده است.

فایل سیستمها

لینوکس طیف گسترده‌ای از فایل سیستمها را پشتیبانی می‌کند. فایل سیستمها در حالت کلی نحوهٔ ذخیره شدن، ایندکس شدن و بازیابی داده‌ها را از دیسک تعریف می‌کنند. متداولترین فایل سیستم موجود برای لینوکس، فایل سیستم ext2 می‌باشد. اگر یکی از توزیعهای استاندارد لینوکس را در اختیار داشته باشید، این فایل سیستم را در آن پیدا خواهید کرد. به هر حال به دلایلی ممکن است از فایل سیستمهای دیگر استفاده شود. برخی مواقع ممکن است بخواهید دیسکها را بین سیستم-

عاملهای مختلف تسهیم کنید که در این صورت باید از ساختار مشترکی پیروی کنید. ممکن است کارآیی یا قابلیت اطمینانی مدنظر شما باشد که فایل سیستم خاصی را بطلبد. بهتر است برای تصمیم گرفتن در مورد اینکه بهترین فایل سیستم برای شما چه فایل سیستمی می‌تواند باشد، با استفاده از برنامه‌های کاربردی خاص، یکسری تست انجام دهید. در زیر چند فایل سیستم مهم آورده شده است:

EXT

http:// oss.software.ibm.com /developer/opensource/ifs

فایل سیستمهای journaling ، درجه بالایی از انعطاف‌پذیری و قابلیت اطمینان برای سیستم‌عاملها فراهم می‌کنند.

حین فعالیت عادی سیستم‌عامل، مقادیر زیادی عملیات ذخیره‌سازی و بازیابی دیسک رخ می‌دهد. از آنجاییکه این اعمال با سرعت نسبتاً کمی صورت می‌گیرند، همهٔ سیستمها از تکنیکهایی برای سرعت بخشیدن به این اعمال بهره می‌گیرند و معمولاً این کار با تعبیه حافظه‌های موقتی[1] در حافظه اصلی سیستم صورت می‌گیرد. به هر حال اگر حین این عمل مشکلی در سیستم پیش آید، ممکن است صحت داده‌های روی دیسک نقض شود. فایل سیستمهای journaling بر این مشکل که اغلب به عنوان یکی از ضعفهای سیستم‌عاملها بشمار می‌رود، فائق آمده است.

XFS

http://www.namesys.com

هانس رایزر طراح فایل سیستم ReiserFS است که یک فایل سیستم journaling با کارآیی بالا تحت GPL می‌باشد. در بحث مجوز دوگانه ذکر شد که رایزر مجوز استفاده از فایل سیستم خود را به عرضه‌کنندگان سیستم‌عاملهای تجاری داده است. این ویژگی امکان استفاده از فناوری فایل سیستم یکسان و تسهیم حافظه بین لینوکس و گونه‌های دیگر یونیکس را فراهم می‌کند.

---

[1] cache

پشتیبانی سخت افزار

امکان لیست کردن انجمنهای پشتیبانی کننده همه دستگاه‌های سخت‌افزاری امکان ندارد. بهترین راه برای اطمینان از پشتیبانی یک توزیع لینوکس از یک دستگاه خاص، بررسی لیست سخت‌افزارهای پشتیبانی شده توسط آن توزیع است [که در سایت شرکت توزیع‌کننده آن موجود است.]

در فصل پنج، توزیعهای لینوکس را با تفصیل بیشتری مورد بررسی قرار خواهیم داد و همچنین در مورد نحوهٔ پشتیبانی توزیعها از دستگاه‌های سخت‌افزاری بحث خواهیم کرد.

خدمات وب وکارگزارهای برنامه‌های کاربردی

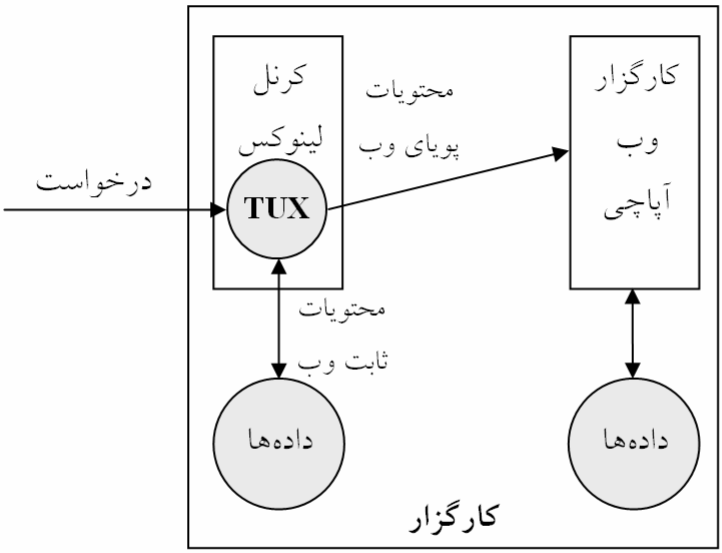
قبلاً در مورد نقش وب سرور آپاچی در رشد چشمگیر لینوکس بحث کردیم. ولی آپاچی، تنها یک نرم‌افزار نیست و کار آن منحصر به یک کارگزار وب ساده نمی‌شود. وب از ارائه محتوای ایستا مثل نشر و چاپ به ارائه محتوای پویا که بیشتر شامل تعاملات پایگاه داده‌ای می‌باشد، گسترش یافته است. علاوه بر این، شاهد تکامل و گسترش تجارت الکترونیکی، منطق تجارت و اخیراً خدمات وب نوینی هستیم. انجمنهای متن‌بازی وجود دارند که پیشنهادهایی برای این فضای درحال رشد و توسعه ارائه می‌دهند. لازم به یادآوری است که همه آنها منحصر به لینوکس نیستند؛ متن‌باز برای سیستم‌عامل‌های غیرلینوکس نیز راه‌حلهای زیادی ارائه کرده است. در اینجا به برخی از آنها نگاه گذرایی خواهیم انداخت:

Apache
http://www.apache.org

آپاچی به صورت یک شرکت غیرانتفاعی (نه به قصد به دست آوردن سود مالی) به منظور پیش برد توسعهٔ هستهٔ سرور وب و همهٔ خدمات پویایی که حول و حوش آن شکل می‌گیرند و توسعه می‌یابند، بنیانگذاری شد.

در سایت این انجمن تعداد زیادی زیرانجمن می‌یابید که پا به پای فناوریهای جدید و در حال گسترش در زمینه وب و سرورهای برنامه‌های کاربردی، تکامل یافته‌اند. یکی از این زیرانجمنها،

انجمن XML برای آپاچی است. XML قالب داده‌ای مشترکی برای توسعه کارگزارهای برنامه‌های کاربردی برای نسل بعدی این کارگزارها ارائه می‌دهد.



شکل ۴-۱: استفاده از TUX به همراه کارگزارهای وب متداول

TUX

http://www.redhat.com/products/ software/linux/ tux/

TUX یک کارگزار وب مبتنی بر کرنل است. سرورهای وب مانند آپاچی مثل یک برنامهٔ کاربردی روی کرنل لینوکس اجرا می‌شوند. چیزی که TUX را خیلی ویژه و خاص می‌سازد این است که به صورت جزئی از کرنل اجرا می‌شود و در نتیجه می‌تواند از کارآیی بالایی برخوردار باشد و بر کارآیی خدمات لایه‌های بالاتر بیفزاید.

شکل ۴-۱ تصویر ارتباط بین TUX درون کرنل و کارگزار وب آپاچی (خارج کرنل) را نشان می‌دهد. ترکیب TUX ین کارگزار وب برای بهبود عملکرد سروکارگزارها بسیار حائز اهمیت است.

JBOSS

http://www.jboss.org/

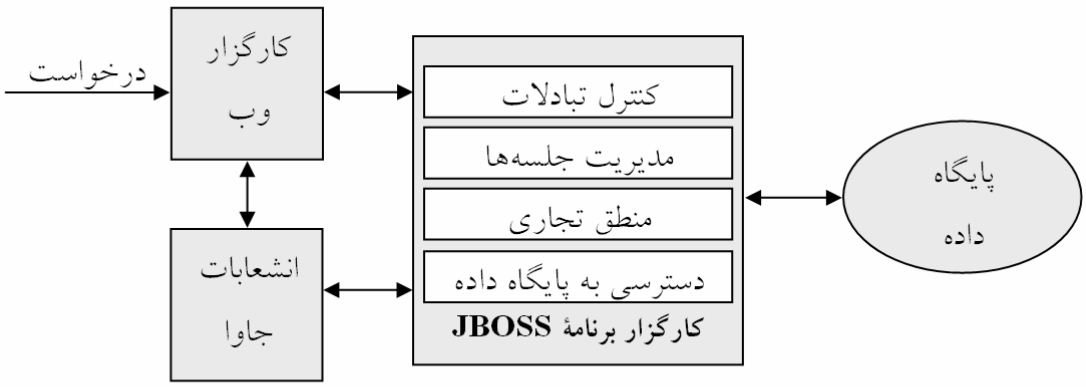
JBOSS نسخهٔ حرفه‌ای جاوای ۲ (J2EE) که بصورت متن‌باز پیاده‌سازی شده است می باشد. J2EE نوعی فناوری کارگزارهای برنامه کاربردی است. پیاده‌سازی‌های تجاری J2EE عبارتند از iPlanet از SUN، Weblogic از BEA و Websphere از IBM. به واسطهٔ رعایت مجوزهای SUN ،

JBOSS نمی‌تواند به عنوان نسخه تأیید شدهٔ J2EE، صاحب مجوز شود. اگرچه از طریق کار با بنیاد آپاچی، SUN در نظر دارد که به تدریج پیاده‌سازیهای متن‌باز J2EE را آزاد کند.

کارگزارهای برنامه‌های کاربردی به طور چشمگیری قابلیتهای وب کارگزارهای سنتی را افزایش داده‌اند. برخی از این قابلیتها شامل خدمات تراکنش برای اطمینان از کنترل همزمانی، خدمات منطق تجارت برای امکان ساخت سریع برنامه‌های کاربردی با ترکیب اجزاء از پیش ساخته‌شده و خدمات پایگاه داده‌ای برای دسترسی به داده‌های شرکت، می‌شوند.

با رشد سریع اینترنت، توسعهٔ برنامه‌های کاربردی زیادی مبتنی بر فناوریهای کارگزارهای برنامه‌های کاربردی صورت گرفت. مبنای دنیای کارگزارهای برنامه‌های کاربردی یونیکس/لینوکس اغلب فناوری جاوا و مبنای دنیای ویندوز، فناوری مایکروسافت .NET (دات نت) می‌باشد.

شکل ۴-۲ ارتباط بین یک کارگزار وب سرور، سرور برنامه کاربردی و پایگاه داده را نشان می‌دهد. بسته به پیچیدگی برنامه کاربردی، کارگزارهای برنامه کاربردی متن‌باز می‌توانند بستر مطمئنی برای پیاده‌سازی‌های تجاری فراهم کنند. در بخش سوم، هنگام بررسی مدلهای تجاری، نگاه عمیق‌تری به تأثیر متن‌باز بر مدل‌های تجاری سنتی خواهیم انداخت.



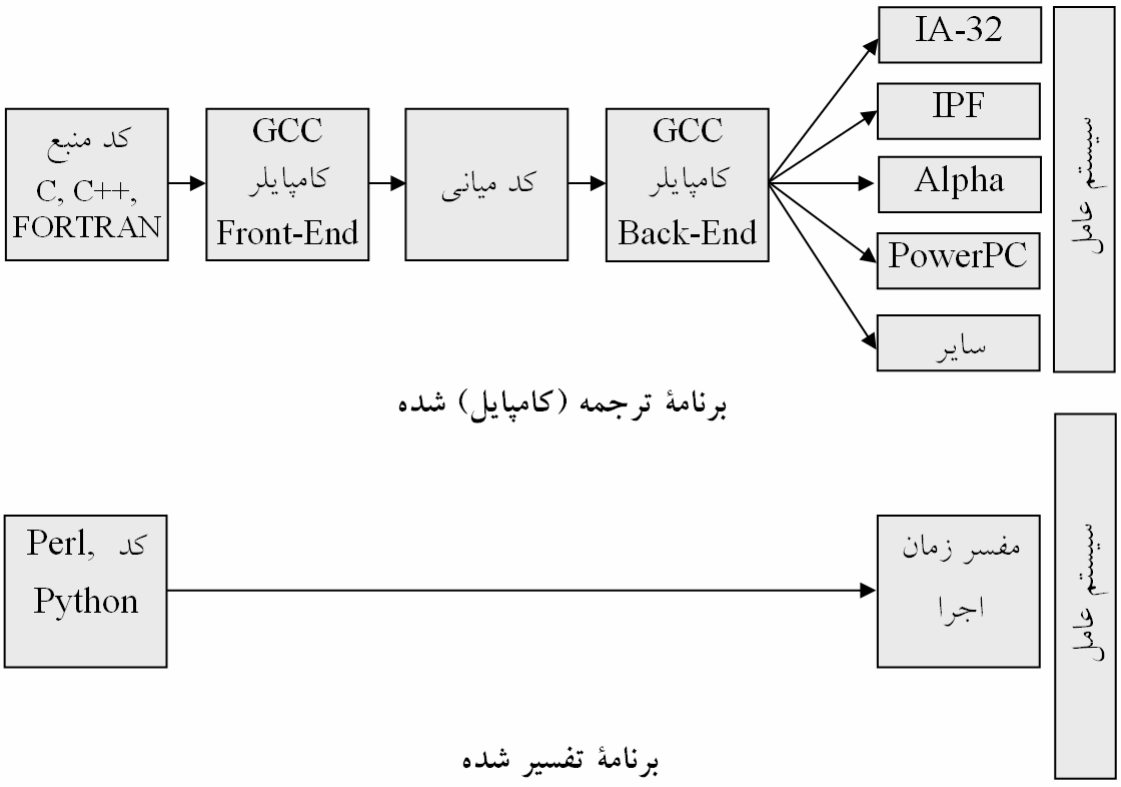
شکل ۴-۲: یک کارگزار برنامهٔ کاربردی متصل به کارگزار وب و پایگاه داده

زبانهای برنامه‌نویسی

تعدادی زبان برنامه‌نویسی متن‌باز وجود دارد که بعضی از آنها تفسیری (حین اجرای برنامه خط به خط به کد ماشین ترجمه می‌شوند) و بعضی از آنها کامپایلری (قبل از اجرا به کد ماشین ترجمه می‌شوند) هستند.

شکل ۴-۳ اختلاف نحوهٔ اجرای زبانهای تفسیری و کامپایلری را روی سیستم نشان می‌دهد:

قسمت عمدهٔ کرنل لینوکس- دستگاه گردانها و مؤلفه‌های دیگر کرنل- با زبان برنامه‌نویسیC نوشته شده است. به هر حال بیشتر نرم‌افزارهای تحت وب و اسکریپت‌ها، مبتنی بر زبانهای تفسیری مثل Perl و Python هستند. در مورد ارتباط دادن این زبانها به طور خاص به لینوکس دقت کنید. این زبانها و مفسر/کامپایلرهای آنها، روی سیستم‌عاملهای متعددی (مثل پیاده‌سازی‌های مختلف یونیکس و ویندوز) موجود هستند.



شکل ۴-۳: برنامه‌های ترجمه شده و تفسیر شده

GCC

http://gcc.gnuorg

احتمالاً GCC، مهمترین ابزاری بود که به لینوس توروالدز امکان نوشتن کرنل لینوکس را داد. GCC، مجموعه‌ای از کامپایلرهایی است که برای ترجمهٔ زبانهای سطح بالا به زبان ماشین به کار می‌روند. فناوری کامپایلر در علم کامپیوتر خیلی مهم است، چرا که می‌تواند نقش مهمی در کارآیی برنامهٔ کاربردی ایفا کند. کامپایلرها، عمدتاً به مؤلفه‌های front-end و back-end تقسیم می‌شوند. مؤلفهٔ front-end کامپایلر، زبانهای سطح بالامثل C، C++، فرترن یا موارد دیگر را به زبانهای میانی ترجمه می‌کنند. مؤلفهٔ back-end کامپایلر، کار سنگین اصلی را انجام می‌دهد. این مؤلفهٔ مسئول دریافت کد میانی و انجام دو وظیفهٔ عمده است: (۱) ترجمه به زبان ماشین و (۲) بهینه سازی. ترجمه به زبان ماشین امکان اجرا شدن برنامه کاربردی شما را روی سیستم مقصد را می‌دهد. این ترجمه به شکلی است که برنامه می‌تواند روی معماری ریز پردازنده مقصد اجرا شود و با سیستم‌عامل سازگار باشد. روشهای بهینه‌سازی کد، موضوع تحقیقات و مطالعات علوم پیشرفته کامپیوتر هستند. در بهینه‌سازی، کد برای اطمینان از اجرا با حداکثر سرعت و کارآیی ممکن توسط پردازندهٔ مقصد، ساماندهی می‌شود.

GCC قسمت اصلی پروژهٔ گنو است. امروزه از کامپایلرهای GCC در طیف عظیمی از سیستم‌عاملها استفاده می‌شود. در واقع ISVها (تولیدکنندگان نرم‌افزارها) برای تضمین کردن قابلیت اجرا بر روی سکوهای مختلفی که پشتیبانی می‌کنند، GCC را برای طراحی نرم‌افزار مورد نظر خود انتخاب می‌کنند. هدف اولیه طراحی GCC این بوده است که ابزار برنامه‌نویسی مشترکی وجود داشته باشد تا بتوان به راحتی در سیستمهای غیرمتجانس مختلف برنامه‌نویسی کرد.

به هر حال GCC در برخی موارد کارآیی مورد نیاز روی یک سکو را ندارد. در نتیجه شرکتهای دیگر کامپایلرهایی ارائه کرده‌اند که برای یک سکوی خاص بهینه شده است. اگر کارا بودن برنامهٔ کاربردی برایتان اهمیت زیادی دارد، استفاده از کامپایلرهای تجاری اغلب نتایج بهتری دربردارد. ولی این، احتمالاً به قیمت از دست دادن قابلیت انتقال تمام شود. اگر درگیر توسعهٔ هستهٔ لینوکس هستید یا می‌خواهید سکوهای متعددی را تحت پوشش قرار دهید، GCC بهترین انتخاب است.

GCC توسط شرکت Cygnus -که مایکل تایمن مؤسس آن بود- تجاری شد. او یکی از اولین مدلهای تجاری متن‌باز را ارائه داده است. اگر چه GCC متن‌باز است، Cygnus آن را خرید و برای آن پشتیبانی و خدمات ارائه داد. Cygnus اکنون به Red hat پیوسته است.

Perl

http://www.perl.org/

Perl که مخفف Practical Extraction Report Language به معنای "زبان استخراج گزارش عملی" است در حقیقت یک زبان برنامه‌نویسی تفسیری مختص پردازش متن است. Perl از سال ۱۹۸۷ مورد استفاده قرار گرفته است و طرفداران زیادی داشته و در نتیجه متخصصین زیادی هم دارد. از این زبان اغلب برای مدیریت سیستم، آپاچی و برنامه‌های کاربردی تحت وب استفاده می‌شود. از آنجایی که برنامه‌های کاربردی تحت وب اغلب با پردازش اطلاعات مبتنی بر متن سروکار دارند، این زبان رویکرد و راه‌حل مناسبی برای این گونه کاربردها است. اگر نگاهی به سازمان خود بیاندازید، احتمالاً تعدادی برنامه‌نویس آشنا با Perl پیدا خواهید کرد. چون Perl برای طیف وسیعی از سیستم‌عاملها قابل دسترس است رویکردهای مناسبی مستقل از سکو برای فعالیتهای توسعه‌ای فراهم می‌کند.

Python

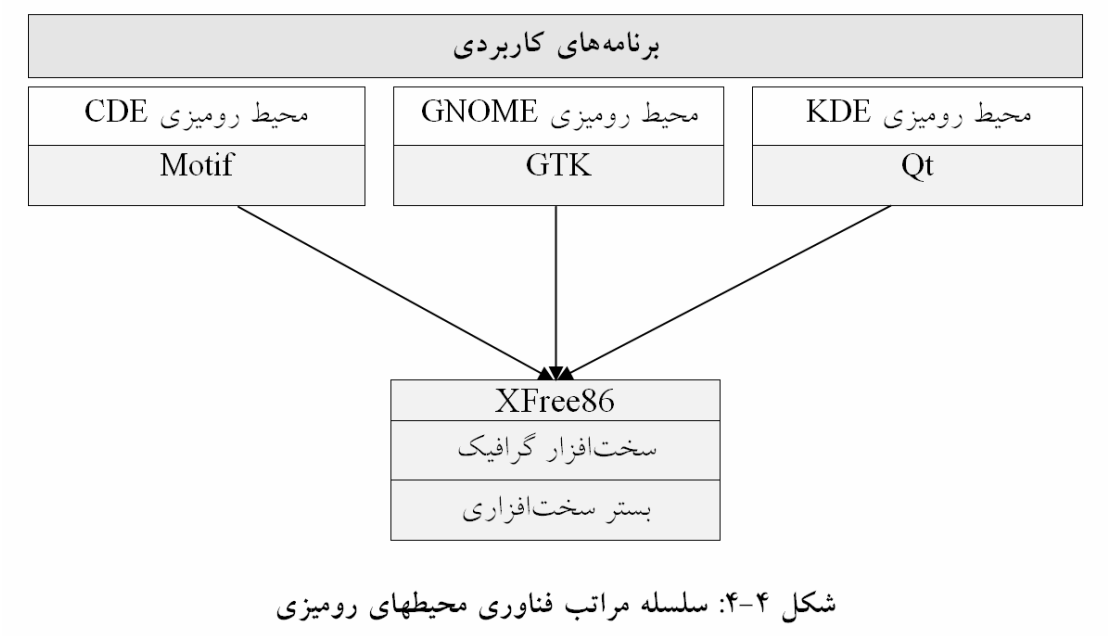
http://www.python.org/

Python نیز یک زبان تفسیری شیء گرا می‌باشد که تحت سکوهای مختلفی موجود است. Python به اندازه Perl طرفدار ندارد، ولی زبانی است که توسعه‌دهندگان شیءگرا مثل کسانی که با C++ آشنا هستند به سرعت یاد می‌گیرند. این زبان نیز با پردازش متن سروکار دارد، ولی در عین حال گزینهٔ مناسبی برای نمونه‌سازی سریع کاربردهای مبتنی بر واسطه‌های گرافیکی است. از Python نیز مانند Perl اغلب در برنامه‌های کاربردی تحت وب استفاده می‌شود.

محیطهای رومیزی و برنامه‌های اداری

یکی از بزرگترین تفاوتهای بین لینوکس و ویندوز وجود چند میزکار[1] گرافیکی در لینوکس و یک میزکار در ویندوز است که توسط مایکروسافت ارائه می‌شود. بعضی معتقدند که این تعدد محیطهای رومیزی باعث سردرگمی می‌شود و هزینه‌های اضافی آموزش و پشتیبانی تحمیل می‌کند. بعضی هم بر این باورند که این پدیده ابتکار و نوآوری بوجود آورده است.

هر دو میزهای کار لینوکس جوان هستند و زندگی خود را از سال ۱۹۹۷ آغاز کرده‌اند. شکل ۴-۴ نحوهٔ تعامل محیطهای رومیزی با کتابخانه گرافیکی XFree86 را نشان می‌دهد.



شکل ۴-۴: سلسله مراتب فناوری محیطهای رومیزی

همچنین پروژه‌هایی به منظور ارائه نرم‌افزارهایی، مشابه مجموعه نرم‌افزارهای اداری مایکروسافت (MS Office) به شکل متن‌باز شکل گرفته‌اند که مشهورترین آنها OpenOffice می‌باشد.

XFree86
http://www.xfree.org/

XFree86، اصلی‌ترین سیستم نمایش پنجره‌ها است که در همهٔ سیستمهای لینوکس از آن استفاده می‌شود. این سیستم، پیاده‌سازی متن‌باز چارچوب کاری X11 است که توسعهٔ آن از MIT[2] آغاز

---

[1] Desktop

[2] موسسه فناوری ماساچوست

شد. XFree86 ، امکان دسترسی سطح پایین به سخت‌افزار گرافیک نصب شده روی سیستم را فراهم می‌کند. اگر از ایستگاه‌های کاری یا میزکارهای لینوکس استفاده می‌کنید، سخت‌افزار گرافیک نصب شده روی سیستم شما باید از XFree86 پشتیبانی کند. اگر چه امکان نوشتن برنامه‌های کاربردی کاملاً کارآ برای X11 وجود دارد ولی پیچیدگیهایی به همراه دارد که باعث می‌شود این کار به ندرت انجام شود. در عوض از کتابخانه‌های نمایش پنجره سطوح بالاتر که با X11 در تعامل هستند، استفاده می‌شوند. کتابخانه‌ای که برای سالهای زیادی مورد استفاده قرار می‌گرفت Motif بود. محیط رومیزی که به همراه Motif استفاده می‌شد، CDE بود. در مورد لینوکس، توسعه‌دهندگان نیاز به کتابخانه‌های نمایش پنجره و محیطهای رومیزی متن‌باز داشتند. برای نیل به این هدف دو پروژهٔ عمده شکل گرفت: GNOME وKDE .

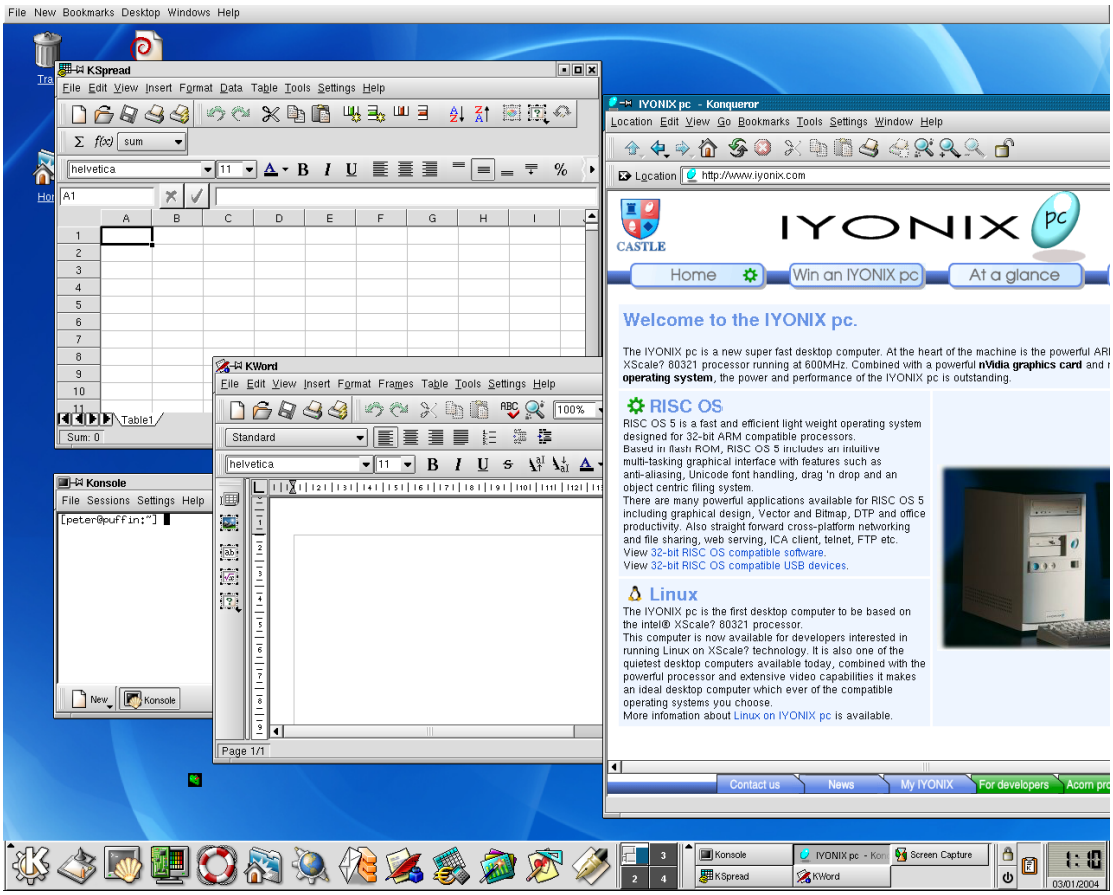
GNOME

http://www.gnome.org/

پروژه میزکار گنوم اواخر تابستان سال ۲۰۰۰ توسط مؤسسه GNOME کلید خورد. گنوم توسط شرکت Ximian (Helox Code سابق) تجاری شد. میزکار گنوم توسط Miguel de Icaza برای ارائه میز کار کاملاً متن‌باز در سال ۱۹۹۷ آغاز شد. هر میزکار مبتنی بر مجموعه‌ای از کتابخانه‌های واسط گرافیک کاربری (GUI) است.

در آن زمان، KDE مبتنی بر کتابخانه‌ای بود که تماماً متن‌باز نبود. کتابخانهٔ نمایش پنجره برای نوشتن برنامه‌های کاربردی گنوم، GTK و برای KDE ، Qt نام دارد. شکل ۴-۵ نمایش صفحه‌ای از میزکار گنوم است.

مؤسسه گنوم با همکاری Ximian، Red hat، HP، IBM وSUN توسعهٔ میزکار گنوم را رهبری می‌کند. این پروژه سکوی توسعهٔ گنوم برای برنامه‌نویسان وGNOME Office را نیز شامل می‌شود. بسیاری از عرضه‌کنندگان یونیکس نیز میزکار گنوم را به عنوان میزکار اصلی محیطهای ایستگاه کاری یونیکس خود انتخاب کرده‌اند. اگر مایلید میزکار همگونی در سیستم‌های یونیکس و لینوکس خود داشته باشید، گنوم انتخاب خوبی خواهد بود.



شکل ۴-۵ محیط رومیزی (میزکار) KDE

شکل ۴-۶ محیط رومیزی (میزکار) گنوم

KDE
http://www.kde.org/
http://www.kdeleague.org/

KDE نیز محیط رومیزی مشهوری برای لینوکس است. بحث درباره اینکه کدام میز کار بهتر است، اغلب سلیقه‌ای است. در واقع این بحث بیشتر برمی‌گردد به اینکه کاربر کدام را ترجیح می‌دهد یا چه کاری می‌خواهد انجام دهد یا یک مشخصه خاصی که در یکی موجود است و در دیگری وجود ندارد. شکل ۴-۶ نمایش صفحه‌ای از میزکار KDE است.

همان گونه که گنوم مؤسسه گنوم را به منظور گردهم آوردن رهبران صنعتی برای کمک و سرمایه -گذاری در فرآیند توسعه گنوم، بنا نهاد، لیگ KDE نیز با اهدافی مشابه تشکیل شد.

Mozila
http://www.mozilla.org/

موزیلا یک مرورگر وب است که وارث مرورگر نت اسکیپ (Netscape) است. موزیلا درحقیقت اسم رمز اصلی مرورگر نت اسکیپ بود و وقتی شرکت نت اسکیپ تصمیم به متن‌باز کردن این محصول گرفت از اسم رمز آن استفاده کرد.

این گونه عرضه نت اسکیپ را می‌توان بارزترین مثال از عرضه یک محصول تجاری تحت مجوز متن‌باز دانست. این کار درسهای خوبی به جامعه متن‌باز داد. چند سال زمان لازم بود تا موزیلا بتواند کارآیی و قابلیت اعتمادی را که ما در نت اسکیپ و مرورگر اینترنت مایکروسافت سراغ داشتیم، بدست آورد.

با اقدام مایکروسافت به عرضه IE[1] به عنوان یک برنامه رایگان همراه ویندوز، عرضه موزیلا تسریع پیدا کرد. این رویدادها همزمان بود با انتشار مقاله معروف اریک ریموند با عنوان "کاتدرال و بازار"همزمان بود.

شرکت نت اسکیپ بر این باور بود که می‌تواند از روش توسعه اشتراکی که اریک در مقاله‌اش توصیف کرده است، برای پیشبرد توسعهٔ یک مرورگر جایگزین استفاده کند. اما این ایده نت‌اسکیپ

[1] Internet Exploner

درست از آب درنیامد. آنها مرتکب چندین اشتباه شده بودند (مثل عرضه کدی که کاملاً کارآ نیست). قسمت سوم این کتاب تماماً در مورد پیشگیری از این خطاها و خطاهای دیگر است.

امروزه موزیلا در زمرهٔ مرورگرهای وب مطرح و قابل اعتماد قرار دارد.[۱]

Open Office
http://www.openoffice.org/

اگرچه چندین پروژهٔ طراحی مجموعه نرم‌افزارهای اداری متن‌باز درحال توسعه است، پروژهٔ OpenOffice قابل اطمینان‌ترین جایگزین برای مجموعهٔ مایکروسافت آفیس شناخته شده است. این پروژه پیاده‌سازی متن‌باز محصول Star Office از شرکت SUN Microsystems است که امروزه موجود می‌باشد.

در سال ۲۰۰۰ میلادی شرکت SUN به منظور ارائه میزکار اداری برای سیستمهای یونیکس و رقابت با مایکروسافت، شرکت StarDivison را خرید و بصورت زیرمجموعه خود درآورد. SUN محصول StarOffice را بصورت محصولی ثبت شده و پشتیبانی شده ارائه می‌کند درحالی که OpenOffice یک سیستم کاملاً آزاد است.

پایگاه داده‌ها

پایگاه داده‌ها زیربنای اصلی بسیاری از برنامه‌های کاربردی را تشکیل می‌دهد. این سیستمها توابعی برای ذخیره‌سازی، سازمان‌دهی و بازیابی اطلاعات به شکل بهینه ارائه می‌دهند. وقتی نام پایگاه داده‌ها را می‌شنویم نامهای Oracle، Sybase و DB2 به یاد می‌آوریم. گرچه این پایگاه داده‌های تجاری، مقیاس‌پذیری فوق‌العاده، کارایی و ویژگیهای زیادی را فراهم می‌کنند، سیستمهای متن‌بازی هم وجود دارند که این نیازها را به شکل بهتر و با قیمت کمتر، برآورده می‌کنند.

[1] امروزه بنیاد موزیلا تبدیل به مهد چندین پروژه متن باز شده است، از جمله Thunderbird که یک نرم‌افزار ایمیل است. این بنیاد اولین نسخه مرورگر موزیلا را نیز تحت نام فایرفاکس در سال ۲۰۰۴ عرضه کرد.

PostgreSQL
http://www.postgresql.org/

PostgreSQL یک موتور پایگاه دادهٔ حرفه‌ای و شیءگرای مبتنی بر SQL است که سابقهٔ آن به اواخر دههٔ ۱۹۷۰ و اوایل دههٔ ۱۹۸۰ در دانشگاه برکلی برمی‌گردد. اجداد قبلی آن با پیاده‌سازیهای تجاری شرکتهای Ingres (که بعدها به Computer Associates پیوست) و Informix (که به IBM پیوست) شکل گرفتند. با درنظر داشتن چنین سابقه‌ای، در اکثر کاربردهای پایگاه دادها می‌توان بطور جدی روی PostgreSQL حساب کرد. برخی شرکتها پشتیبانیهای تجاری برای این پایگاه داده‌ها ارائه کرده‌اند.

MySQL
http://www.mysql.org/

MySQL پایگاه داده معتبر متن‌باز دیگری است که اغلب در برنامه‌های کاربردی تحت وب و کاربردهای شخصی از آن استفاده می‌شود. به هر حال MySQL پایگاه داده‌ای با کارایی بالا است که طرفداران زیادی دارد و خود را به عنوان پراستفاده‌ترین پایگاه دادهٔ متن‌باز معرفی می‌کند. افراد زیادی را به سمت خود جلب کرده است.

این پایگاه داده همچنین مثال عملی خوبی است از مجوزدهی دوگانه. همه می‌توانند پایگاه داده MySQL را با رعایت شرایط GPL به کار گیرند. به هر حال اگر شما نیاز به یک مجوز تجاری برای محصول خود دارید و قصد دارید پایگاه داده‌ای را همراه با محصولتان ارائه دهید، MySQL گزینه مناسبی است.

PDAها (Personal Digital Assistants)

ارائه یک لیست کامل و جامع از همهٔ پروژه‌های متن‌بازی که نرم‌افزارهایی برای سکوهای قابل انتقال طراحی و توسعه می‌دهند، ممکن نیست. اگر به پیرامون خود بنگرید می‌توانید برنامه‌های کاربردی متن‌باز متعددی برای PalmOS، PocketPC و تعداد دیگری از سکوهای لینوکس پیدا کنید. در

اینجا تعدادی از پروژه‌های اصلی متن‌باز که روی لینوکس برای سیستمهای دستی[1] تمرکز کرده‌اند، آورده شده است.

Handheld Linux
http://www.handhelds.org/
جامعه متن‌باز در حال توسعهٔ سیستم‌عاملها و نرم‌افزارهایی برای سکوهای دستی است. بیشتر کاری که امروزه انجام می‌شود بر روی IPAQ متمرکز شده است زیرا IPAQ، سیستم‌عامل خود را که PocketPC OSنام دارد در حافظهٔ فلش ذخیره می‌کند و این امر به توسعه‌دهندگان امکان جایگزینی PocketPC OS را با سیستم‌عامل خودشان می‌دهد. HP نیز بطور غیر رسمی از این گونه کارهای توسعه‌ای متن‌باز حمایت می‌کند.

صنعت به سرعت در حال انتشار جایگزینهای PDA مبتنی بر این روش توسعهٔ متن‌باز است. حتی تعدادی توزیع کاملاً توسعه‌یافته برای این سکوها تهیه شده است.

کلاسترها (Clusters)

از کلاسترها برای تغییر مقیاس سیستمها بصورت افقی، برای رسیدن به کارآیی بیشتر و فراهم کردن درجهٔ بالاتری از دسترس‌پذیری برای کل سیستم استفاده می‌شود. شرکتهایی وجود دارند که کلاسترهای متن‌باز را پشتیبانی می‌کنند و پیاده‌سازیهای مختلف کلاستر را به شکل تجاری در می‌آورند.

کلاستر Beowulf، برجسته‌ترین راه‌حل در زمینه کلاسترهای لینوکس، برای محاسبات سنگین، می‌باشد.

Beowulf
http://www.beowulf.org/
پروژهٔ Beowulf مثال فوق‌العاده‌ای از نحوهٔ تأثیر اقتصاد کالایی[2] بر دنیای انفورماتیک است. هدف این پروژه فراهم کردن امکان محاسبات سنگین با استفاده از محصولات کالایی می‌باشد

---

[1] Handheld

[2] Commodity

. Bewulf، راه‌حلی نرم‌افزاری برای کلاستربندی تعداد کثیری از گره‌ها[1] جهت محاسبات موازی ارائه می‌دهد.

اگر درگیر پروژه‌هایی هستید که می‌توانند از معماریهای محاسبه موازی بهره‌گیرند، Beowulf نقطهٔ فوق‌العاده‌ای برای آغاز کار است.

سازمانها

رشد و شهرت لینوکس و جنبش متن‌باز مدیون سازمانهای زیادی است. اغلب این سازمانها، شرکتهای غیرانتفاعی هستند که قصدشان بدست آوردن سود مالی نیست و وقت و انرژی خود را به ارتقاء جنبه‌های مختلف توسعه و گسترش متن‌باز اختصاص داده‌اند. تعداد کثیری از این سازمانها با سرمایه و بودجهٔ اندکی کار می‌کنند و نیازمند کمک و حمایت هستند. کمکها می‌تواند به صورت اهداء تجهیزات، منابع، خدمات بازاریابی و مبالغ نقدی باشد.

Linux International
http://www.Li.org

Linux International، سازمانی غیرانتفاعی است که لینوکس را در سطح بین‌المللی ترویج می‌دهد و توسط جان هال[2] مدیریت می‌شود. این سازمان، لینوکس را به رهبران تجاری می‌شناساند و مزایای لینوکس را برای استفاده‌های تجاری عنوان می‌کند. همچنین منابع مالی پروژه‌هایی را که نزد جامعه لینوکس ارزش و اهمیت بیشتری دارند، تأمین می‌کند.

بنیاد نرم‌افزار آزاد (Free Software Foundation)
http://www.fsf.org

همان طور که قبلاً اشاره شد، هدف FSF، ترویج و ارتقاء نرم‌افزار آزاد برای عموم است. FSF زاییدهٔ ذهن ریچارد استالمن و منزلگاه پروژه گنو و مجوز GNU GPL است. اگر می‌خواهید نسخه‌ای از متن GPL را در اختیار داشته باشید یا از جزئیات نرم‌افزار رایگان کسب اطلاع کنید، باید به اینجا مراجعه کنید.

---

[1] node

[2] John “Maddog” Hall

Open Source Initiative
http://www.opensource.org/

OSI در حقیقت OSD [1] را ترویج و ارتقاء داده و فهرستی رسمی از مجوزهای مورد قبول و مصوب متن‌باز را نگهداری می‌کند. در زمان نوشتن این کتاب، چیزی بیش از ۳۰ مجوز تصویب شده وجود دارد. با توجه به اینکه طراحی مجوز جدید امکان پذیر است، قبل از اینکه مجوز متن‌باز جدیدی طراحی کنید بهتر است نگاه دقیقی به مجوزهای موجود بیاندازید.

آزمایشگاه توسعه متن‌باز (Open Source Development Lab)
http://www.osdl.org/

OSDL، توسط جمعی از رهبران صنعتی با انگیزه ارتقاء لینوکس برای مراکز داده‌ها و ارتباطات راه دور تشکیل شد. OSDL، آزمایشگاه پیشرفته‌ای در اختیار دارد که به هرگونه منابع و ابزاری -که ممکن است برای تست سنگین یک پروژهٔ متن‌باز نیاز باشد- مجهز است. هر فردی می‌تواند برای فرستادن پروژه خود به آزمایشگاه و دسترسی به منابع آن درخواست کند.

OSDL برآوردی از کارآیی درخواستی افراد به عمل می‌آورد و همواره تلاش می‌کند بر این اساس تجهیزات خود را تکمیل کند. اگر در زمینه تسریع و شتاب‌دهی به آماده‌شدن قابلیتهای لینوکس برای مراکز داده و محیطهای ارتباطات راه دور انگیزه دارید، OSDL انجمن خوبی برای آشنا شدن است.

گروه استانداردهای آزاد (FSG)
http://www.freestandardsorg /

فصل هفتم جزئیات کاملی از استانداردهای ارائه شده توسط این گروه را شرح می‌دهد. FSG در حال حاضرگروه‌های کاری اصلی زیر را شامل می‌شود:

- پایه استانداردهای لینوکس (LSB)

LSB، مجموعه‌ای از استانداردهای واسط برای اطمینان از قابل انتقال بودن روی سکوهای متن‌باز را تعریف می‌کند.

- بین‌المللی سازی یا Li18ux

[1] Open Source Definition

Li18ux در زمینه ایجاد استانداردهایی برای ترویج استفادهٔ لینوکس در بازارهای بین‌المللی تلاش می‌کند.

کنسرسیوم لینوکس توکار
http://www.embedded-linux.org /
لینوکس به سرعت در کاربردهای توکار محبوبیت پیدا کرد و امروزه تعدادی از شرکتها و توزیع‌های لینوکس فعالیت خود را روی لینوکس توکار متمرکز کرده‌اند.

کنسرسیوم لینوکس توکار سازمانی تجاری است که به ارتقاء لینوکس برای کاربردهای توکار اختصاص دارد. اگر پروژه‌ای توکار را شروع کرده‌اید و لینوکس نیازهای کاری شما را برآورده می‌کند یا در جستجوی شرکتهای دیگر برای کار کردن با آنها هستید، از این سازمان بازدید کنید.

پروژهٔ مستند سازی لینوکس
http://www.linuxdoc.org
واژه‌ای که اغلب در این انجمن به کار می‌رود، "HowTo" است. همان طور که از نام آن برمی‌آید، HowTo راهنماهایی است که توسط افراد مختلف جامعه، در زمینه استفاده از لینوکس، نوشته شده است و در مورد هر موضوعی که فکر کنید مطلبی تهیه شده است. پروژهٔ مستندسازی لینوکس تلاشی برای رسمی کردن مستندات لینوکس است.

همه می‌توانند در این پروژه شریک باشد. این پروژه در زمینه ترجمهٔ راهنماهای موجود نیز فعالیت می‌کند.

جمع بندی

وقتی شما مطالعه این فصل را به پایان می‌رسانید باید کاملاً متقاعد شده باشید که متن‌باز یک رویکرد گذرا نیست. آنقدر تعداد شرکتها، گروه‌ها و افرادی که در ارتباط با آن هستند زیاد است که امکان کنار گذاشتن آن نیست. حرکتی که باید انجام شود، تضمین آینده‌ای روشن برای طیف وسیع برنامه‌های کاربردی متن‌باز موجود است و هرچه تعداد آنها بیشتر شود سهم همتای تجاری آنها نیز از بازار کمتر می‌شود.

اگر شما این رده از برنامه‌ها را در محیط فنآوری اطلاعات خود به کار می‌برید، باید نگاه جدی به این موضوع داشته باشید. اگر در هر یک از این زمینه‌ها توسعه‌دهنده هستید یا رقبای جدیدی دارید که انگیزه‌های اقتصادی یکسانی با شما ندارند، اینجاست که شما باید به جای رقابت با آنها از مزایای آنها استفاده کنید و مدل تجاری خود را به سطح جدیدی برسانید. قسمت سوم کتاب چگونگی این کار را به شما خواهد آموخت.

این فصل را می‌توان آخرین بخش آموزش مقدماتی نیز تلقی کرد. اکنون شما اطلاعات کافی برای چگونگی گسترش لینوکس و متن‌باز در سازمان خود دارید. قسمت دوم کتاب تماماً در مورد به کارگیری آنچه که تا اینجا یاد گرفتید، در کسب و کار است .

قسمت دوم

لینوکس در عمل

با اطلاعاتی که تاکنون در مورد لینوکس به دست آورده‌اید، می‌توانید این مبحث را آغاز کنید که پیاده‌سازی و گسترش لینوکس در سازمان و تجارت در عمل به چه معنی است . لینوکس یک پروژه متن‌باز خیلی بزرگ است. اگر چه این قسمت روی گسترش جنبه‌های لینوکس تاکید دارد، با این حال قسمت عمده آن قابل اعمال به دیگر نرم‌افزارهای متن‌باز نیز هست.

قسمت دوم با نگاه مفصلی به توزیعهای لینوکس آغاز می‌شود. توزیعها برای ارائه یک سیستم‌عامل مفید و کارا، روی کرنل لینوکس ساخته می‌شوند. بعد، نگاهی به تاثیرات استفاده از لینوکس و متن‌باز در تجارت خواهیم انداخت و همان طور که پیش می‌رویم استانداردهای مرتبط با لینوکس را مرور می‌کنیم . قسمت دوم با بررسی اجزاء عملی به کارگیری، انتقال از یک سکو یا سیستم‌عامل به سکو یا سیستم‌عامل دیگر، هم زیستی[1] ، پشتیبانی و آموزش به پایان می‌رسد.

[1] coexisense

فصل پنجم:

توزیعها-تکمیل لینوکس

اگر در حال تهیه خودرویی هستید که با آن از نقطه A به نقطه B بروید، هیچ وقت تنها موتور خودرو را نمی‌خرید. دریافت و ساخت کرنل لینوکس را می‌توان مشابه در اختیار داشتن موتور دانست. بدیهی است که شما نیاز به یک خودروی کامل دارید و بسته به نیازتان ممکن است خودرو سواری، وانت و یا هر وسیله‌ای از بی شمار وسیله نقلیه موجود را انتخاب کنید. حتی وقتی به خوبی می‌دانید که چه می‌خواهید، ممکن است امکانات اضافه‌ای را فراتر از نیاز خود در خودرو سفارش دهید.

توزیعهای لینوکس مشابه آن اتوموبیل هستند. توزیعها کرنل لینوکس را با قابلیتها و ویژگیهای دیگر ترکیب می‌کنند. برخی از توزیعها مختص کاربردهای گسترده موجود در بازار هستند و برخی دیگر بر نیازهای مکانهای جغرافیایی و یا محلهای خاص تمرکز می‌کنند. یکی از پیش‌نیازهای اصلی برای درک لینوکس، فهم جزئیات و ساختار یک توزیع است. در این فصل خواهید آموخت:

- یک توزیع چیست
- بسته‌ها و وابستگیها
- عرضه‌کنندگان اصلی توزیع‌ها
- چه چیز یک توزیع را از توزیع دیگر متفاوت می‌کند
- مواردی که باید هنگام انتخاب یک توزیع لحاظ شوند
- وابستگی به یک توزیع
- نگاهی گذرا به استانداردهای مرتبط با توزیعها

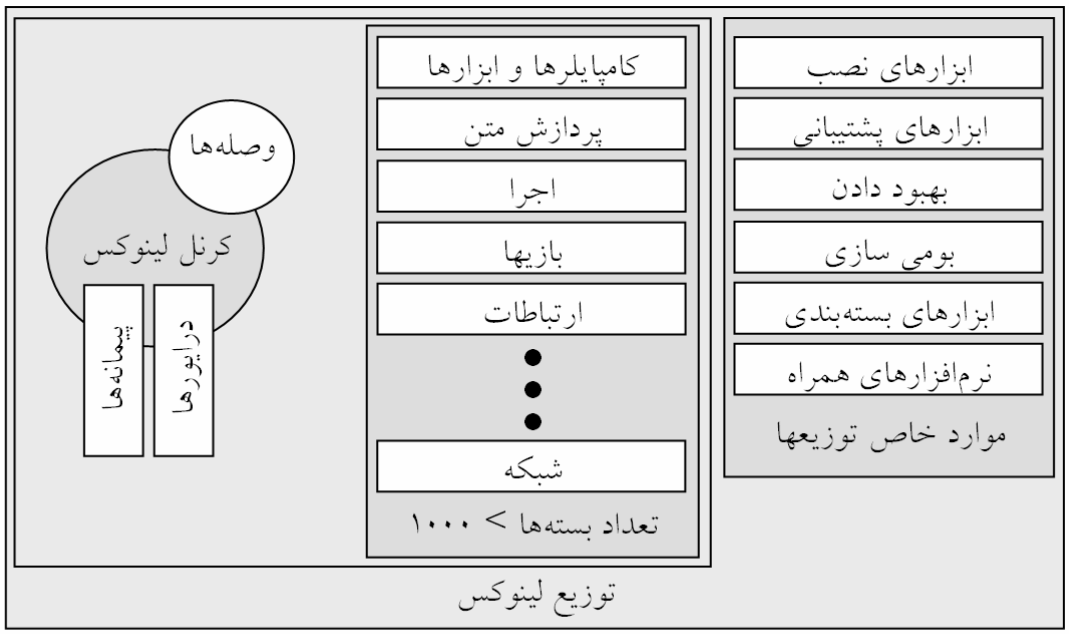
آنچه که شما در شرکت خود به کار می‌گیرید و گسترش می‌دهید، توزیعهای لینوکس هستند. اگر برنامه‌های کاربردی تولید می‌کنید، درک تفاوتهای میان توزیعها تضمین خواهد کرد که شما بتوانید تمامی محیطهای ممکن را پشتیبانی کنید.

توزیع لینوکس

اکثر مردم وقتی به "لینوکس" اشاره می‌کنند، عموماً منظورشان توزیعهای لینوکس است. همان طور که به زودی خواهید آموخت، یک توزیع، توسط یک عرضه‌کننده تهیه می‌شود. آنها مؤلفه‌ها و اجزای متعددی را با کرنل همراه می‌کند. عرضه‌کنندگان توزیع عموماً وظایف زیر را انجام می‌دهند:

- یک نسخه خاص کرنل را انتخاب می‌کنند.
- وصله‌هایی (Patch) را به منظور افزودن کارآیی یا حل مشکلات طراحی می‌نویسند یا انتخاب می‌کنند.
- در زمینه استفاده از مجموعه خاصی از بسته‌های نرم‌افزاری و نسخه‌های آنها تصمیم‌گیری می‌کنند.
- کارهای ارزش افزوده (کارهایی که باعث افزایش ارزش توزیع می‌شود) خود را به آن می‌افزایند مثل نرم‌افزار نصب بهینه و توسعه‌ابزارها.
- کل مجموعه را در یک رسانه (مثلاً لوح فشرده) جهت پخش ترکیب می‌کنند.
- هر جزء و همچنین کل توزیع را تست و تایید نموده و از کارآیی آن اطمینان حاصل می‌کنند.
- در برخی موارد از برنامه‌های کاربردی شرکتهای ثالث استفاده می‌کنند.
- از نشانه‌ها و علامت تجاری خاص خود در توزیع استفاده می‌کنند.
- در مورد خط مشیهای معرفی به بازار تصمیم‌گیری می‌کنند.
- خدمات ارزش افزوده خود را همراه با توزیع ارائه می‌دهند.

هر توزیع لینوکس با دیگری فرق دارد. شکل ۵-۱ دیدی از ساختاری که تقریباً در همه توزیعها مشترک است، نشان می‌دهد. همان طور که می‌بینید، یک توزیع خیلی بیشتر از فقط یک کرنل است.



شکل ۵-۱: شمای کلی یک توزیع لینوکس

عرضه‌کنندگان توزیع کارهای بسیار زیادی برای تهیه یک لینوکس انجام می‌دهند. اینکه آنها این کار را چقدر خوب انجام می‌دهند و چقدر خوب می‌توانند برای کارهایشان پشتیبانی ارائه دهند، فاکتورهای اصلی در انتخاب یک توزیع، برای نصب می باشد.

اغلب عرضه‌کنندگان توزیع، مجموعه‌ای از بسته‌های نرم‌افزاری متن‌باز را با هم ترکیب می‌کنند. این بدین معنی است که توزیعهای آنها را می‌توان به صورت رایگان دریافت و نسخه‌برداری کرد. بنابراین عرضه‌کنندگان توزیعها مجبور شدند مدلهای تجاری خود را به منظور حفظ پیشرفت شرکت، تکامل بخشند. اینجا خلاصه کوتاهی از نحوه کسب درآمد عرضه‌کنندگان توزیع آمده است:

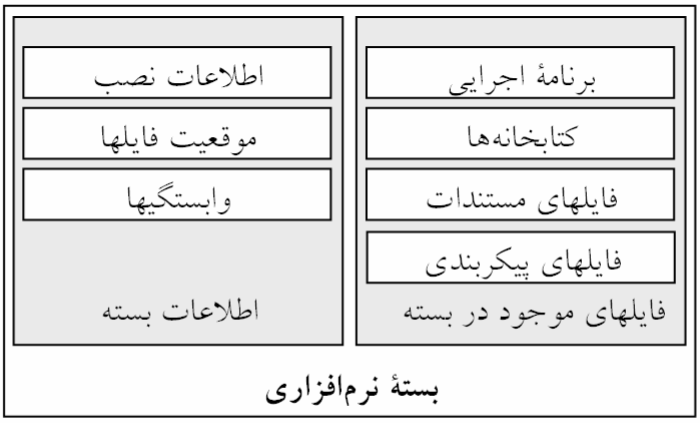
- فروش مجموعه رسانه و مستندات در یک جعبه از طریق کانالهای خرده فروشان
- ارائه خدمات پشتیبانی و به‌روزرسانی
- ارائه خدمات حرفه‌ای مرتبط با توزیع لینوکس خود

- دریافت هزینه از تولیدکنندگان تجهیزات که توزیع را با محصولات نرم‌افزاری یا سخت-افزاری خود ارائه می‌کنند.

عرضه‌کنندگان توزیع به طور پیوسته مدلهای تجاری خود را بهبود می‌بخشند و بر فهرست زیر بدون شک افزوده خواهد شد. در قسمت سوم کتاب به جزئیات مدلهای تجاری لینوکس و متن‌باز خواهیم پرداخت.

بسته‌های نرم‌افزاری

هر توزیع نماد "بسته نرم‌افزاری" را به همراه دارد، زیرا یک توزیع همه بسته‌های نرم‌افزاری تعیین شده برای آن نسخه را گردآوری می‌کند. هر بسته از یک یا چند فایل تشکیل شده که در حالت کلی یک وظیفه (مثل یک کامپایلر، یک مرورگر ،یک کارگزار وب یا غیره) را انجام می‌دهد.



شکل ۵-۲: ساختار عمومی یک بستهٔ نرم‌افزاری

برای روشن‌تر شدن موضوع باید اشاره کرد که این نگهدارنده بسته است که درمورد چیدمان بسته تصمیم می‌گیرد. برخی بسته‌ها بسی‌ار بزرگ هستند و شامل هزاران فایل را می‌شوند در حالی که برخی دیگر خیلی کوچکند و از بیش از یک یا دو فایل تشکیل نمی‌شوند. شکل ۵-۲ طرح کلی یک بسته را نشان می‌دهد.

همان طور که می‌بینید، یک بسته هم فایلهایی مورد نیاز برای به انجام رساندن وظایف خود و هم اطلاعاتی را که مورد نیاز مدیر بسته برای نصب بسته روی سیستم مقصد است را در برمی‌گیرد.

فهرست جدول ۵-۱ از بسته پایدار Debian گرفته شده است. دیگر عرضه‌کنندگان توزیع ممکن است این فهرست را به اشکال دیگری گروه‌بندی کنند. بدیهی است که فهرست کردن همه بسته‌ها ممکن نیست (فقط Debian از حدود ۹۰۰۰ نرم‌افزار تشکیل شده است) ولی جدول ۵-۱ دسته-بندی اصلی بسته‌ها را فهرست کرده است.

جدول ۵-۱ دسته بندی نرم‌افزارهای بسته‌ها

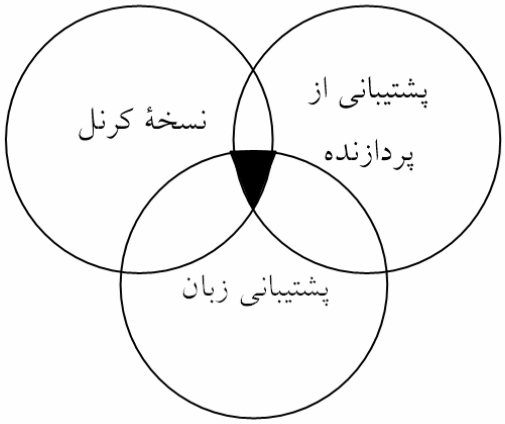
نام دسته	شرح
برنامه‌های مدیریتی	برنامه‌های کمکی برای راهبری منابع سیستم، مدیریت کاربران و انجام سایر امور راهبری سیستم
برنامه‌های ارتباطی	نرم‌افزارهای ارتباطی از طریق یک مودم سنتی
توسعه و برنامه‌نویسی	ابزار و محیطهای برنامه‌نویسی، کامپایلرها و کتابخانه‌ها
مستندات	سؤالهای متداول (FAQ) راهنماها و غیره
ویرایشگرها	نرم‌افزارهای ویرایش فایلها
الکترونیکی	نرم‌افزارهای الکترونیکی
بازی	سرگرمی
گرافیک	ویرایشگرها، نمایش‌دهنده‌ها و مبدلها
رادیوی آماتوری	نرم‌افزارهای کار با رادیوهای آماتوری
مفسرها	هرنوع مفسری برای زبانهای تفسیرشونده و پردازشگرهای ماکروها
کتابخانه‌ها	کتابخانه‌ها برای کار برنامه‌ها مورد نیاز هستند و امکانات خوبی در اختیار برنامه‌نویسان قرار می‌دهند.
نامه	برنامه‌های مسیردهی، خواندن و نوشتن پیامهای ایمیل
ریاضیات	نرم‌افزارهای ریاضی
متفرقه	برنامه‌های کمکی متفرقه که در دسته خاصی نمی‌گنجند
شبکه	دیمونها (برنامه‌های کارخواه که در پس زمینه اجرا می‌شوند) وکارگزارها (مانند telnet و مرورگر وب) که لینوکس را به دنیای خارج متصل می‌کنند.
گروه‌های خبری	نرم‌افزارهای دسترسی به یوزنت وکارگزاهای News

نام دسته	شرح
نرم‌افزار محدود شده توسط قانون یا حق ثبت	اکثر بسته‌های نرم‌افزاری را می‌توان در ایالات متحده استفاده کرد ولی قابل صادر کردن نیستند (حتی نمی‌توان مجدداً آنها را وارد کشور کرد.) برخی از این نرم‌افزارها ممکن است به خاطر مسائل مربوط به حق ثبت، در ایالات متحده و یا کشورهای دیگر قابل استفاده نباشند. باید با دقت محدودیتهای نرم‌افزارها را مطالعه و درصورت لزوم با وکیل خود مشورت کنید.
کتابخانه‌های قدیمی	نسخه‌های قدیمی کتابخانه‌ها که برای سازگاری با نسخه‌های قبلی برنامه‌ها حفظ می‌شوند.
سیستم‌عاملها و فایل سیستمهای دیگر	نرم‌افزارهای اجرا کننده برنامه‌های سایر سیستم‌عاملها و استفاده از فایل سیستم آنها
پوسته‌ها	پوسته‌های فرمان رابطهای کاربرپسند برای افراد مبتدی
صوتی	نرم‌افزارهای کمکی مدیریت صدا: میکسرها، اجرا و پخش‌کننده‌ها، ضبط صوت‌ها
TeX	نرم‌افزار معروف تایپ و برنامه‌های مربوط با آن
پردازش متن	برنامه‌های کمکی قالب بندی و چاپ اسناد متنی
برنامه‌های کمکی	برنامه‌های کمکی برای کار با دیسک/فایلها، ابزار بایگانی و ساختن نسخه پشتیبان، نظارت بر عملکرد سیستم و سیستمهای ورودی
نرم‌افزار وب	سرورهای وب، مرورگرها، پراکسی‌ها و ابزار دریافت فایل
نرم‌افزار سیستم X Window	کارگزارهای X، کتابخانه‌ها، فونتها، مدیریت پنجره‌ها، برنامه‌های ترمینال و برنامه‌های دیگر در این زمینه

هر بسته‌ای نگهدارنده یا گروهی از نگهدارندگان دارد و از نسخه‌های مختلفی تشکیل می‌شود. اکثر بسته‌ها بومی‌سازی شده‌اند و می‌توان آنها را به هر زبانی یافت. عرضه‌کنندگان تولید نه تنها در مورد اینکه چه بسته‌هایی باید با توزیعشان ارائه شود، بلکه در مورد نسخه‌های بسته‌ها، وصله‌های همراه با این بسته‌ها و زبانهایی که باید پشتیبانی شوند، نیز تصمیم می‌گیرند.

یک عرضه‌کننده توزیع برای اینکه بسته‌ای را با یک توزیع همراه کند دقت می‌کند که آن بسته، نسخه کرنل مورد استفاده را پشتیبانی کند، نسخه بسته برای پردازنده‌ای که پشتیبانی می‌شود موجود باشد و زبان انتخابی هم پشتیبانی گردد. این گزینه‌ها، اصلی‌ترین موارد برای در نظر گرفتن

و لحاظ شدن هستند. اگر چه ممکن است موارد دیگری هم در برخی شرایط در نظر گرفته شود. نمودار ون در شکل ۵-۳ شرایط مورد نیاز برای اینکه یک بسته در توزیع قرار گیرد را نشان می‌دهد.



شکل ۵-۳: دیاگرام وِن مشترکات بسته‌ها

همیشه باید به خاطر داشته باشید که یکی از ویژگیهای کلیدی متن‌باز این است که شما به عنوان مشتری کنترل را در دست دارید نه عرضه‌کننده. اگر از تصمیماتی که فراهم‌کننده توزیعتان گرفته است راضی نیستید، می‌توانید هر اندازه تغییر که مایل هستید روی آن اعمال کنید. اگر بسته خاصی مورد نظرتان است که در توزیع ارائه نشده است، می‌توانید آن را در اینترنت پیدا کنید و یک نسخه از آن را برای نصب دریافت کنید.

قالب و فرمت بسته

بسته‌ها از ترکیب فایلها ساخته می‌شوند و با به کارگیری ابزارهای ویژه‌ای که پیمانه¬های خاصی تولید می‌کنند، آماده استفاده می‌شوند. دو قالب مشهور بسته وجود دارد:

یکی توسط Red Hat طراحی و توسعه‌یافته است که Red hat Package Manager یاRPM نامیده می‌شود و دیگری متعلق به انجمن Debian است و از آن به فرمت DEB (یا دات دب) یاد می‌شود.

عرضه‌کنندگان توزیعها

تعداد عرضه‌کنندگان لینوکس چنان زیاد است که نمی‌توان همه آنها را نام برد. اگر بخواهید فقط توزیعهای خاص یا مناسب نیازهای محلی را بشمارید، تعداد آنها به صدها عدد می‌رسد. در اینجا روی توزیعهایی که فراگیرتر هستند (سهم عمده‌ای از بازار را در اختیار دارند) تاکید می‌شود و مثالهایی از توزیعهای که بر بازارها یا مکانهای جغرافیایی خاص و محدود تمرکز دارندّ، نیز آورده خواهد شد. می‌توانید فهرست کاملی از توزیعها را در وب سایت *linuxorg* مشاهده کنید.

توزیع‌های فراگیر

Red Hat

در گوشه و کنار جهان رِدهَت شناخته شده‌ترین توزیع لینوکس است و یکی از چند شرکت موجود تجاری است که کاملاً روی لینوکس تمرکز دارد. باب یانگ[۱] و مارک ایوینگ[۲]، ردهت را در سال ۱۹۹۴ تشکیل دادند. ردهت یکی از اولین نمونه‌هایی بود که نصب و استفاده از لینوکس را آسان کرده بود. ردهت، سهم عمده‌ای در بازار دارد ولی در همه مناطق جهان نفوذ نکرده است.

ردهت از توزیع خود به‌عنوان یک کارت اعتباری معتبر استفاده می‌کند که دربردارنده سابقه آن در خدمت به لینوکس است. آنها تعدادی از توسعه‌دهندگان را -که به طور مستقیم با لینوکس و دیگر اجزاء سیستم‌عامل سروکار دارند- به کار می‌گیرند و در زمینه‌های بسیاری از قبیل RPM، ابداعاتی کرده‌اند.

وقتی مدلهای تجاری را با جزئیات بیشتری بررسی می‌کنیم، خواهید دید که ردهت، اکثر انرژی خود را صرف توسعه خدمات پشتیبانی تجاری (غیررایگان) کرده است. آنها از همان ابتدای کار دریافتند که با فروش مجوز نرم‌افزارهای آزاد نمی‌توانند درآمد مناسبی برای ادامه کار شرکت ایجاد کنند. با این وجود، ردهت نسبت به متن‌باز وفادار مانده است و از همراه کردن و عرضه هرگونه

---

[1] Bob Young

[2] Marc Ewing

نرم‌افزاری که منطبق با تعریف متن‌باز نیست، امتناع ورزیده است. معلوم نیست تاچه زمانی می‌توانند با این روش ادامه دهند و تنها زمان، مشخص کننده این مسأله است.

همچنین به یاد آوردی که ردهت، Cygnus Solution را در اختیار خود درآورد که نقش کلیدی نگهدارنده ابزارهای کامپایلر GCC را بر عهده دارد.

ترکیب یکی از مشهورترین توزیعهای لینوکس با مجموعه‌ای از خدمات و پشتیبانی، ردهت را به نقش آفرینی با اعتبار در دنیای لینوکس تبدیل کرده است.

SUSE

این توزیع که در آلمان پایه‌ریزی شده است، توزیعی است که می‌تواند روی طیف وسیعی از معماریهای ریزپردازنده استفاده شود. SUSE بواسطه پایه آلمانی‌اش، در منطقه اروپا از محبوبیت خاصی برخوردار است[1]. این توزیع، اغلب رهبر فناوری لینوکس خوانده می‌شود و می‌تواند هم در محیطهای کارگزار و هم در محیطهای کارخواه به کار گرفته شود و به عنوان سکوی اصلی توسعه بسیاری از تولیدکنندگان نرم‌افزار استفاده شود. ردهت، اصلی ترین رقیب آن محسوب می‌شود. برخی جنبه‌های مدل تجاری آن خیلی شبیه به ردهت است در حالیکه در برخی جنبه‌ها کاملاً متفاوت است. تمرکز SUSE بر پشتیبانی چندین سکو و دسته بندی و فروش مجدد نرم‌افزارهای شرکتهای ثالث به همراه لینوکس باعث شده است که مدل تجاری آن به گونه دیگری رشد کند. این مسأله برای آنها جریان درآمدی از جانب تولیدکنندگان تجهیزات اصلی (OEM) و مجوزهای نرم‌افزار ایجاد کرده است.

Caldera

Caldera، روند تکامل جالبی در دنیای لینوکس داشته است. این شرکت که خود را وقف لینوکس کرده بود، به SCO (از تولیدکنندگان یونیکس) ملحق شد و توانست به پشتوانه قوی‌تری برای فعالیت خود تکیه کند. اکنون caldera، هم محصولات لینوکس و هم محصولات یونیکس تولید می‌کند و روی مسائل اساسی یکپارچه‌سازی بین این دو رویکرد تمرکز کرده است. caldera در

[1] توضیح مترجم: شرکت SuSE در ۱۳ ژانویه ۲۰۰۴ توسط شرکت آمریکایی ناول خریداری شد.

حال توسعه محصولات ارزش افزوده برای ارائه به همراه توزیع خود است تا بتواند علاوه بر خدمات حرفه ای و پشتیبانی، از مجوز توزیع خود هم درآمد کسب کند.

Turbo Linux

این توزیع بیشترین نمود را در بازارهای آسیایی داشته است و اگر چه Red Hat در این زمینه سایه به سایه Turbo Linux پیش می‌رود و تهدیدی برای آن به شمار می‌رود، با این حال هنوز هم Turbo Linux در این بازار قدرت اول است. یکی از ویژگیهایی که Turbo Linux با توزیع خودش ارائه داده است توسعه زبان آسیایی است که همین مسأله باعث محبوبیت آن شده است. Turbo Linux، تاکنون چندین مدل تجاری مختلف را امتحان نموده که از جمله آنها می‌توان به تلاش ناکام این شرکت در ادغام با Linux Care اشاره کرد.

اکنون Turbo Linux یک مدل تجاری دارد که بر ارائه راه‌حلهایی در زمینه قابلیت مدیریت، کلاستربندی و پایگاه داده‌ها تمرکز دارد. به علاوه این شرکت توانسته است توزیع خود را برای چندین ریزپردازنده مختلف آماده کند.

Mandrake

Mandrake حیات خود را با شعار "یک ردهت بهتر" شروع کرد. این شرکت کار خود را با یک نسخه از توزیع ردهت آغاز کرد و ویژگیهای مفیدی را به آن افزود و یک توزیع -که برای آخرین نسل پردازنده‌های اینتل کامپایل و بهینه شده بود- تهیه کرد. Mandrake ، با گذشت زمان از ردهت فاصله گرفت. امروزه این توزیع به عنوان یک لینوکس رومیزی قوی مطرح است و تلاش می‌کند جای خود را در میان سیستمهای کارگزار به عنوان رویکردی قابل اطمینان باز کند. Mandrake در فرانسه پایه‌ریزی شده که این امر آن را در میان البته بعد از ردهت اروپائیان محبوب ساخته است.

Debian

Debian بواسطه نداشتن ارتباطات تجاری، توزیعی قابل توجه است. این توزیع، تنها توزیعی است که ۱۰۰٪ مبتنی بر انجمنها است. ایان مرداک[1] مؤسس Debian است. نام Debian تلفیق نام همسرشDebby و نام خودشIan است.

[1] Ian Murdoc

این خصیصه که Debian مبتنی بر ارتباطات تجاری نیست، Debian را میان رهبران و توسعه - دهندگان جامعه متن‌باز محبوب ساخته است. انجمن Debian شامل صدها توسعه‌دهنده است که جهت عمده فعالیت آنها در راستای دیگر توزیعهای فراگیر است. همچنین به یاد آورید که پیشینه OSD نیز از قرارداد اجتماعی Debian (*debian.org/social_contract*) می‌باشد.

Debian به این می‌بالد که نسبت به سایر توزیعها بیشترین سکو را پشتیبانی می‌کند. شرکتهای زیادی مایل به همکاری با Debian هستند، زیرا در اینجا فرآیند توسعه هیچ زمانبندی ندارد و کارهای عرضه شده به محض حاضر شدن پذیرفته می‌شوند و از آنجایی که همه چیز باز و آزاد است در محیطی برابر رقابت می‌کنند. اگر عرضه‌کننده نرم‌افزار هستید و نیاز به استانداردسازی مؤلفه‌های خاص زمینه کاری خود هستید، Debian می‌تواند بهترین محیط برای کارکردن باشد.

بسیاری از انجمنها و سازمانها Debian را به عنوان پایه‌ای برای کارشان به کار می‌گیرند، سپس برای دیگر توزیعها خدمات پشتیبانی فراهم می‌کنند. Debian به واسطهٔ کم کاری در پشتیبانی جدیدترین نوآوریها در بسته‌ها و طراحیهای کرنل مورد انتقاد قرار گرفته است. Debian همواره به خاطر عقب بودن از سایر توزیعها در پشتیبانی از ابداعات جدید در نرم‌افزارها و نسخه‌های جدید کرنل مورد انتقاد واقع شده است. ولی این در واقع شمشیری دو لبه است. گرچه ممکن است Debian تا حدی عقب‌تر از دیگر توزیعها حرکت کند، اما به عنوان توزیعی شناخته شده است که کاملترین آزمایشها روی آن صورت گرفته است و روی گسترده‌ترین طیف از سکوها و با بیشترین پایداری کار می‌کند.

شرکتهایی که درپی بازترین توزیع با حداقل وابستگی به تهیه کننده و حداکثر پایداری هستند Debian را انتخاب می‌کنند. تنها نکته منفی در مورد Debian این است که به واسطهٔ فقدان پشتیبانی شرکتی تولیدکنندگان نرم‌افزارهای تجاری این توزیع را آخرین انتخاب قرار می‌دهند.

United Linux

United Linux (UL) همکاری تعداد زیادی از عرضه‌کنندگان پیشرو توزیعهای لینوکس برای ایجاد یک مبنای استاندارد برای لینوکس یا ایجاد یک لینوکس منطبق بر LSB است (در فصل ۷ در این

مورد بحث خواهد شد.) از آنجایی که ساخت یک توزیع کار پرهزینه‌ای است و برای عرضه‌کنندگان، ارائهٔ محصولات برای توزیعهای متفاوت، مشکل می‌باشد، تعداد زیادی از عرضه‌کنندگان برای ساخت یک توزیع واحد لینوکس گردهم آمدند. آنها می‌توانند با همسوسازی تلاشهای خود هزینه‌های ساخت توزیع پایه را بین هم تقسیم کنند و انرژی خود را بر کارهای باارزش دیگری متمرکز کنند.

در زمان نوشتن این کتاب UL کارهای ابتدایی خود را شروع کرده است و برای پیش‌بینی دربارهٔ این که آیا UL در صنعت به عنوان یک توزیع رایج محبوب خواهد شد یا خیر خیلی زود است. خوشبختانه این تلاشها باعث تبعیت از LSB و سازگاری برنامه‌های کاربردی با این استاندارد خواهد شد.

توزیع‌های محلی

برخی توزیعها نیازهای خاص بازارهای منطقه‌ای را پاسخ می‌دهند. به چندین مورد از آنها به عنوان مثالهایی از این دست اشاره خواهد شد. ولی نمی‌توان فهرست کاملی از این توزیعها ارائه نمود. بسته به جائی که زندگی می‌کنید، ممکن است توزیعی محلی وجود داشته باشد که نیازهای شما را برآورده سازد.

Red Flag

این توزیع می‌کوشد که سیستم‌عامل لینوکس را به کشور بزرگ چین معرفی کند. Red Flag هم لینوکس توکار و هم لینوکس فراگیر را تهیه کرده است. پیشرفت این لینوکس در راستای نیازهای بازارهای آسیای جنوب شرقی می‌باشد.

اگر تولیدکننده برنامه‌های کاربردی لینوکس هستید و تصمیم دارید چین را بازار هدف محصولات خود قرار دهید، لازم است با یک فراهم کننده محلی لینوکس مثل Red Flag ارتباط برقرار کنید.

Connectiva

این توزیع به عنوان یک توزیع مختص آمریکای لاتین شناخته شده است. Connectiva در برزیل پایه‌ریزی شده و به زبانهای اسپانیایی و پرتغالی توزیع می‌شود. Connectiva نیز همانند اغلب تولیدکنندگان لینوکس، یک سری خدمات پشتیبانی و آموزشی ارائه می‌کند که بر کسب وکارهای موجود در بازار آمریکای لاتین تمرکز دارد.

توزیعهای اختصاصی

مثالهای زیر نحوه تمرکز تولیدکنندگان توزیع بر بازارهای خاص را نشان می‌دهند. اکثر این تولید-کنندگان در یک زمینه خاص تبحر دارند.

MSC.Linux

شرکت MSC.Software، در زمینه برنامه‌های کاربردی محاسباتی با کارآیی بالا فعالیت می‌کند و در زمینه کاری‌اش شناخته شده است. توزیع MSC، مختص کاربردهای شبیه‌سازی، کلاستربندی و کارآیی بالا است. MSC به همراه توزیع خود، خدمات سخت‌افزاری، مشاوره و ساختن توزیعهای سفارشی را نیز ارائه می‌کند و در نهایت سعی می‌کند یک سیستم کاری بهینه برای محیطهای کاربردی خاص مشتریان تهیه کند.

MontaVista

MontaVista (که نام اولیه توزیع آنها Hard Hat Linux بود) بر بازارهای لینوکس توکار و ارتباطات راه دور تمرکز دارد و توزیعهایی تهیه می‌کند که قابلیت اجرای برنامه‌های کاربردی روی پردازنده‌های کوچک و توکار را دارد. همچنین با افزودن ویژگیهای پیشرفته به کرنل، امکان ارائه قابلیتهای بلادرنگ را فراهم می‌کند. MontaVista، چندین نوع پردازنده توکار را پشتیبانی می‌کند، محیط توسعه‌ای (برنامه‌نویسی) مبتنی بر کاربردهای توکار فراهم می‌کند. همچنین این شرکت به پشتیبانی از چندین نوع برد کامپیوتر تخصصی (مثل cPCI و VME) مبادرت کرده است.

توزیعهای سیستم‌عاملهای غیرلینوکسی

واژه ″ توزیع″ برای اشاره به هرگونه بسته‌بندی از فایلها، برنامه‌ها، ابزارها و خدمات به کار می‌رود که لزوماً شامل سیستم‌عامل لینوکس نیست. بهترین مثال از این مورد توزیع میزکار گنوم است. گرچه یک توزیع لینوکس اغلب از مجموعه‌ای از بسته‌های نرم‌افزاری تشکیل شده است که میزکار گنوم را نیز شامل می‌شود، اما ممکن است عرضه‌کنندگان تجاری بسته نرم‌افزاری بهبود یافته و ثبت شده‌ای تحت عنوان ″توزیع گنوم″، طراحی و ارائه کنند.

در ادامه مثال میزکار گنوم، Ximian نیز توزیعی طراحی کرده است که موارد زیر را با هم ترکیب می‌کند:

- بسته‌های مورد نیاز برای اجرای گنوم
- ابزارهای پیشرفته و مخصوص توسعه‌دهندگان
- کد منبع
- نرم‌افزارهای اداری
- قابلیتهای به‌روزرسانی نرم‌افزار
- نسخه‌هایی برای سیستم‌عاملهای مختلف(مثل HP-UX و سولاریس)
- رسانه و مستندات

البته این فقط یک مثال است. تعدادی از شرکتهای بازرگانی تلاش در طراحی و ساخت توزیعها برای هر تعداد از اجزای سیستم‌عامل لینوکس دارند. بعضی تلاشها موفق‌تر از بقیه هستند. نکتهٔ کلیدی که باید به خاطر داشته باشید این است که گرچه واژه "توزیع" اغلب در مورد لینوکس به کار می‌رود، اما ممکن است به دیگر مجموعه‌های نرم‌افزارهای کاربردی نیز اطلاق شود.

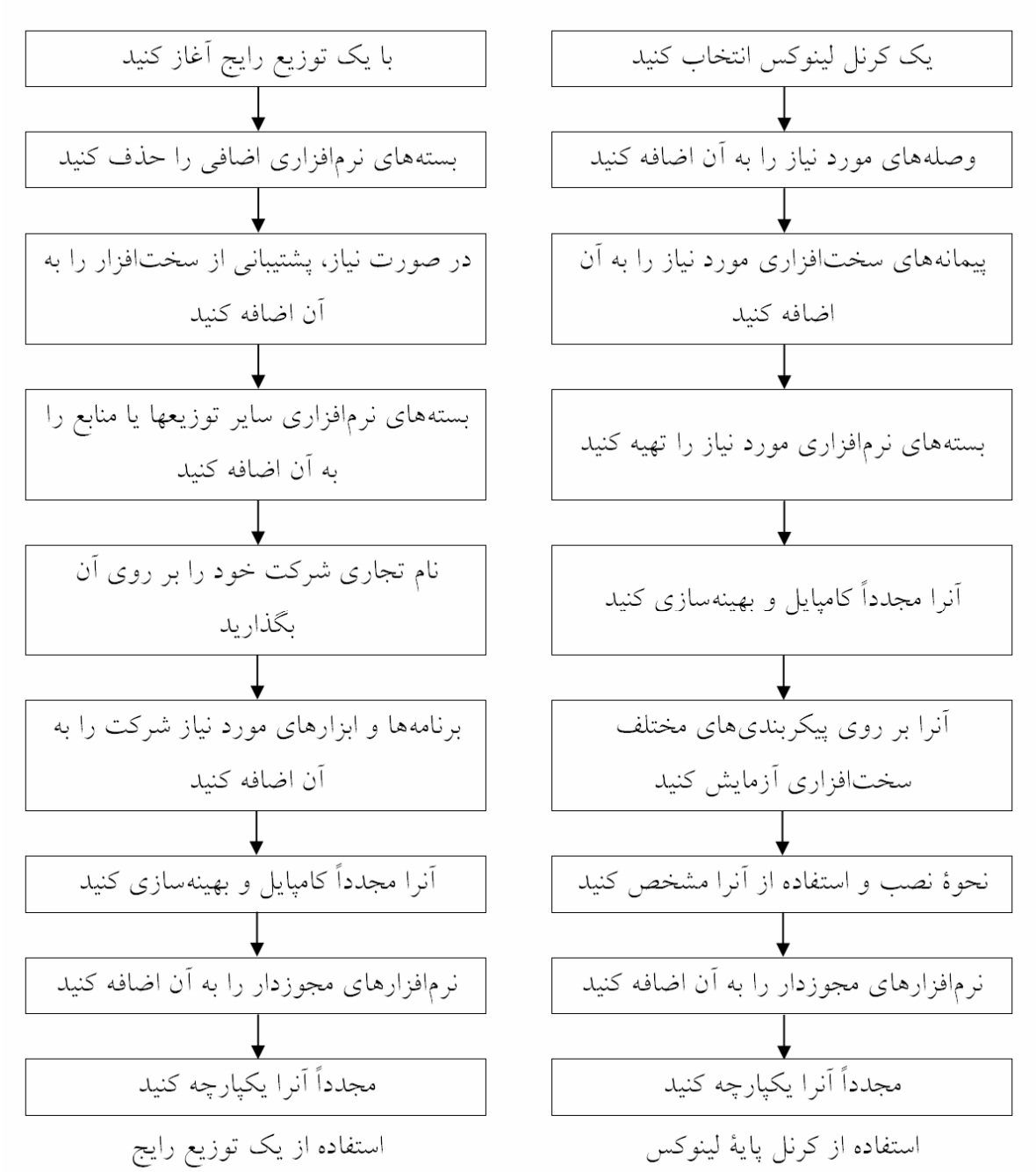
طراحی توزیع برای خودتان

بسیاری از شرکتها به این نتیجه می‌رسند که نیاز به طراحی و ساخت توزیع دلخواه خود برای برآوردن نیازهای خاص سازمان خود دارند. به هر حال فرآیند طراحی یک توزیع نباید ساده گرفته شود. بسته به گستردگی تغییرات و ویژگیهای مورد نظر ممکن است متحمل هزینه زیادی شوید. نوعاً توزیعهای طراحی شده متناسب با نیازها و خواسته‌ها با موارد زیر بنا می‌شوند می‌شوند:

- یک کرنل لینوکس خاص همراه با وصله‌ها و پیمانه¬های از پیش‌تعیین‌شده
- اجزای مربوط به نامهای تجاری ثبت شرکتی
- ابزارها و قابلیتهای خاص شرکت شما
- عاملهای مدیریتی از پیش‌تنظیم‌شده

- برنامه‌های کاربردی از پیش نصب شده که دارای مجوز هستند
- شبکهٔ پیکربندی شده و مؤلفه‌های دیگر
- تمهیدات امنیتی منطبق با استانداردهای شرکت
- قابلیتهای خاص برای تعامل با قسمتهای دیگر سازمان
- پشتیبانی سخت‌افزارهای خاصی که در شرکت استفاده می‌شوند
- حذف همه بسته‌های غیرضروری
- یک کرنل و تعدادی بسته نرم‌افزاری که کامپایل شده و برای اجرا روی سخت‌افزاری خاص بهینه شده‌اند
- چند برنامه نصب‌کننده توزیع حاصل، برای محیطهای مختلفی که در سازمان مورد استفاده هستند (میزکار، کارخواه‌ها، کارخواه‌های چند پردازنده و غیره)

طراحی توزیع خودتان زمانی باید انجام شود که سازمان فناوری اطلاعات شما آن قدر توسعه‌یافته باشد که آمادگی مدیریت تغییرات مداوم همه بسته‌های تشکی‌ل دهنده توزیع را داشته باشد. به مقوله خود اتکایی باید همانند یکی از پروژه‌های نرم‌افزاری سازمان نگریسته شود. شکل ۵-۴ دو روندکار متفاوت برای ساخت یک توزیع دلخواه را نشان می‌دهد.



شکل ۵-۴: ساختن یک توزیع لینوکس

فرآیند سمت چپ با یکی از توزیعهای فراگیر آغاز شده، مؤلفه‌های نامطلوب را حذف می سپس مؤلفه‌های جدید را اضافه می‌کند. فرآیند سمت راست با یک کرنل لینوکس آغاز و توزیع مورد نظر روی آن ساخته می‌شود. اینکه کدام فرآیند را انتخاب کنید بستگی به نیازها و درجه کنترل و تغییرات مورد نظر شما دارد.

پشتیبانی از چندین توزیع

یکی از انتخابهای کلیدی برای شرکتهای بزرگ چند ملیتی تصمیم در مورد این مسأله است که آیا شرکت یک توزیع را برای همه کاربردها پشتیبانی خواهد کرد یا چندین توزیع برای کاربردهای مختلف ارائه خواهد داد.

شما به عنوان یک عرضه‌کننده نرم‌افزار، همواره خواهان هرچه بزرگتر کردن بازار محصولات خود هستید. وقتی تصمیم به گسترش برنامه‌های کاربردی خود روی لینوکس می‌گیرید، ممکن است تلاش کنید توزیعی را انتخاب کنید که با آن بیشترین آشنایی را دارید. در این بین ممکن است یکی از مشتریان شما از شما بخواهد که از توزیع رایج دیگری یا حتی توزیعی که به دلخواه تغییر یافته است، پشتیبانی کنید. استانداردها به این منظور گسترش یافته‌اند که اختلافات بین توزیعها، تأثیر منفی روی عرضه‌کنندگان نرم‌افزار نداشته باشد. شما باید برای تبعیت از آن استانداردها حداکثر تلاش را بکنید. از آنجایی که استانداردها، نقش کلیدی در زمینه طراحی توزیعها و نرم‌افزارها دارند، در فصل هفتم به طور مفصل به آنها خواهیم پرداخت.

روش دیگری که عرضه‌کنندگان نرم‌افزار استفاده می‌کنند، این است که به جای تمرکز بر وابستگی به توزیعها بر وابستگی به مؤلفه‌های یک توزیع تأکید دارند. لیستی از وابستگیها مانند زیر تهیه کنید:

- نسخه کرنل
- وصله‌های مورد نیاز
- بسته‌های مورد نیاز
- نسخه‌های تمامی بسته‌ها
- وابستگیهای بیرونی برنامه‌ها
- نیازمندیهای سخت‌افزاری

این لیست جامع، باعث حذف وابستگیها به یک توزیع پایه می‌شود و امکان پشتیبانی هر توزیعی که را بازار تقاضا کند به شما می‌دهد. این مسأله نیاز شما به آزمایش برنامه‌های کاربردی خود تحت ترکیبات مختلف توزیعها و سخت‌افزارها را مرتفع نمی‌کند، ولی شما را مطمئن می‌سازد که خود را وابسته یک توزیع خاص که فرصتهای بازاری آینده شما را محدود خواهد کرد، نساخته‌اید.

استانداردها

فصل هفتم نگاه مفصلی به استانداردهای مرتبط با لینوکس و توزیعها خواهد داشت. عموماً همه توزیعها مبتنی بر یک کرنل مشترک و مجموعه مشترکی از بسته‌ها هستند، ولی به هرحال هر توزیع ممکن است انتخابهای متفاوتی در زمینه نسخه‌های این مؤلفه‌ها و آنچه که باید در کار شامل شود و یا جزء کار نباشد داشته باشد. اگر تصمیم به اجرای لینوکس روی معماریهای ریزپردازنده غیرفراگیر، دارید، ممکن است بسته‌های کمتری نسبت به لینوکسهای نصب شده در سیستمهای رایج در اختیار داشته باشید.

تعدادی استاندارد در حال توسعه است که تضمین می‌کند توزیعها بتوانند بر ارزشها و کاربردهای خاص خود بیفزایند و درعین حال عرضه‌کنندگان سخت‌افزاری و برنامه‌های کاربردی بتوانند به سادگی و بطور یکسان همه توزیعها را پشتیبانی کنند.

جمع بندی

اکنون شما باید درک روشنی از گزینه‌های پیش رو در زمینه توزیعی که می‌خواهید در محیط کاری خود استفاده کنید، داشته باشید. توزیعها مجموعه‌ای از بسته‌های نرم‌افزاری هستند که در کنار هم قرار می‌گیرند و برای اطمینان از اینکه این مجموعه از نسخه‌ها همگی به خوبی با هم کار می‌کنند، آزمایش می‌شوند. توزیعها امکان نصب و استفاده مجموعه‌ای از نرم‌افزارهای مناسب کسب و کار شما را فراهم می‌کنند.

در فصل بعدی نگاهی به هزینه‌های لینوکس خواهیم انداخت. گرچه می‌توان از توزیعها بدون پرداخت هزینه‌ای استفاده کرد، اما استفاده و پشتیبانی مؤثر آن، در طول چرخه حیات سیستم برای شما هزینه‌هایی دربرخواهد داشت. نحوهٔ محاسبه این هزینه‌ها به طور قابل توجهی با نحوهٔ محاسبه شما در گذشته فرق دارد.

فصل ششم:

هزینه لینوکس و متن‌باز

لینوکس رایگان است، ولی نه ۱۰۰٪. هر مدیر با تجربه‌ای می‌داند که هزینه‌های مؤلفه‌ها تنها بخش کوچکی از کل هزینه‌ها هستند و هزینه کل رویکرد باید با درنظر گرفتن چرخه حیات آن، محاسبه شود. واژه معادل صنعتی که در این زمینه استفاده می‌شود «هزینه کل مالکیت» یا TCO[1] است. ماشینی را که در فصل گذشته می‌خواست شما را از نقطه A به نقطه B ببرد، به خاطر می‌آورید؟ بسته به نوع ماشین، نیاز دارید که هزینه سوخت، بیمه، نگهداری، اجزاء و فاکتورهای دیگر را در طول چرخه عمر این وسیله لحاظ کنید. این امر در مورد تهیه فناوریهایی مثل متن‌باز نیز صادق است، مخصوصاً اینکه در مورد متن‌باز، آزاد بودن آن تعبیر به هزینه نداشتن می‌شود.

در این فصل مؤلفه‌های تأثیرگذار در هزینه‌های متن‌باز را بررسی خواهیم کرد و ممکن است متفاوت از هر محاسبه دیگری (محاسبات مربوط به هزینه) باشد که شما در گذشته داشته‌اید. به خاطر داشته باشید که لینوکس یک پروژه متن‌باز خیلی عظیم است. گرچه ممکن است تمایل داشته باشید در زمینه هزینه‌های مربوط به لینوکس تمرکز کنید، اما مواردی که توضیح داده می‌شوند، مجازاً قابل استفاده در هر پروژه متن‌باز دیگر نیز هستند.

اهداف این فصل عبارتند از :

- فهم و درک هزینه کل یک رویکرد
- تأثیرات متن‌باز روی هزینه‌های رویکرد
- نحوه بدست آوردن نرم‌افزارهای متن‌باز
- چه فرایندهای تجاری نیاز دارند که از نو آزموده شوند

[1] Total Cast Ownership

این فصل اطلاعاتی را در اختیار شما می‌گذارد که هنگام محاسبه هزینه کل مربوط به استفاده و اجرای یک محیط لینوکسی مورد نیاز است.

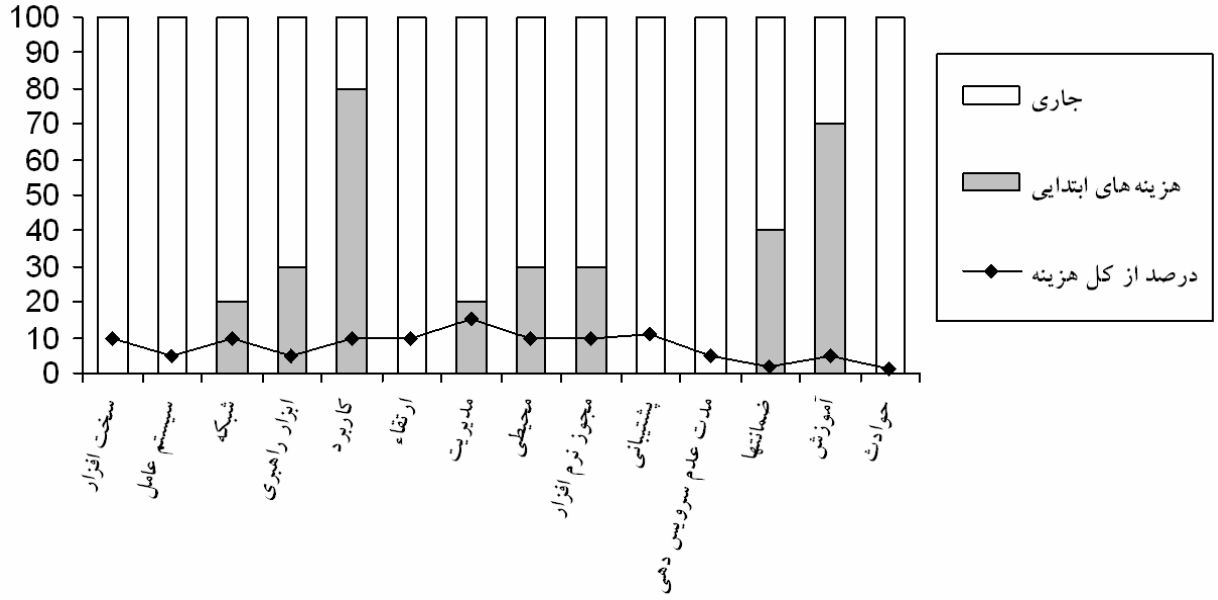
هزینه‌ها

لینوکس و دیگر نرم‌افزارهای متن‌باز نحوهٔ اندازه‌گیری هزینهٔ یک رویکرد را متحول کرده‌اند. در زیر برخی از هزینه‌های سنتی مربوط به رویکردهای مبتنی بر فناوری اطلاعات آورده شده است:

- سخت‌افزار
- سیستم‌عامل
- زیرساخت شبکه
- ابزارهای مدیریت و کمکی
- استفاده
- ارتقاء رویکرد در حال استفاده
- مدیریت و راهبری پیوسته شامل ردیابی مجوزها
- شرایط محیطی (فضای کاری، برق مورد نیاز، تهویه و غیره)
- مجوزهای برنامه‌های کاربردی
- هزینه‌های پشتیبانی (برای همه مؤلفه‌ها)
- خرابیهای پیش‌بینی نشده
- تعهدات و تضمینهای تمدید شده
- آموزش
- حوادث ناگوار و دیگر ریسکهای تجاری

وقتی که محاسبات مربوط به میزان «بازگشت سرمایه» (ROI) را انجام می‌دهید، باید هزینه‌های اولیه و هزینه‌های نگهداری پیوسته را نیز به طور دقیق مشخص کنید. مثلاً هزینه خرید مجوز اولیه یک برنامه کاربردی جدید در قیاس با هزینه‌های پشتیبانی -که این برنامه کاربردی در طول چرخه

حیاتش دارد- می‌تواند ناچیز باشد. از طرف دیگر عرضه‌کنندگان اغلب برخی از این مولفه‌های هزینه را (مثل سخت‌افزار با سیستم‌عامل) با هم عرضه می‌کنند که این کار محاسبه و مقایسه را مشکل می‌سازد. شکل ۶-۱ یک نمونه از هزینه‌های نسبی به کارگیری رویکردهای کامپیوتری در یک سازمان فرضی را بطور نظری نشان می‌دهد.



شکل ۶-۱: نمودار هزینهٔ نمونه در پیاده‌سازی یک راه حل کامپیوتری

چون هر سازمان با سازمان دیگر متفاوت است، شکل ۶-۱ تنها تلاش دارد که یک سناریوی محاسبه هزینه نوعی را ارائه دهد (در سازمان شما ممکن است اساساً متفاوت باشد). این نمودار، درصدهای نسبی هزینه‌های گسترش یک رویکرد، کامل را نشان می‌دهد. نمودار میله‌ای درصد هزینه اولیه تهیه یک مؤلفه از رویکرد را نسبت به هزینه‌های مربوط به نگهداری آن مؤلفه در حین پیشرفت کار، نشان می‌دهد. به عنوان مثال، هزینه‌های شرایط محیطی شامل برخی هزینه‌های اولیه (ساخت یک اتاق کامپیوتر، نصب تجهیزات برق و تهویه و غیره) می‌شود. ولی در عین حال هزینه‌های محیطی دیگری مربوط به نگهداری مؤلفه‌های نیز موجود می‌باشد که در حین پیشرفت کار شکل می‌گیرد. قسمت خاکستری رنگ میله هزینه‌های ثابت اولیه را نشان می‌دهد. قسمت سفیدرنگ میله هزینه‌های در حال پیشرفت را نشان می‌دهد. نمودار خطی درصد نسبی این دو هزینه را برای هر مؤلفه از کل رویکرد نشان می‌دهد.

همان گونه که ملاحظه می‌کنید اکثر مؤلفه‌ها هم هزینه‌های ثابت اولیه و هم هزینه‌های متغیر درجریان کار دارند. در اکثر موارد هزینه‌های در جریان کار، درصد بیشتری از هزینه‌های ثابت را به خود اختصاص می‌دهند. همچنین می‌توان مشاهده کرد که هیچ کدام از مؤلفه‌ها قسمت بزرگی از کل هزینه رویکرد را به خود اختصاص نمی‌دهند. پیچیدگی که در اینجا نشان داده نشده است این است که اکثر سازمانها چندین رویکرد را به کار می‌گیرند و بسیاری از هزینه‌های مؤلفه‌ها با توجه به این مسأله با هم جمع خواهند شد. تحلیل تجاری شما باید این پیچیدگی را به شمار آورد. مهمترین فاکتور این است که اگر در حال استفاده از چندین رویکرد هستید و می‌توانید هزینه‌های ثابت را یک جا کنید، در آن صورت مؤلفه‌های متأثر از متن‌باز می‌توانند تاثیر جدی روی صرفه‌جوئی در هزینه‌ها داشته باشند.

اثر متن‌باز

از آنجایی که شناختن دقیق هزینه‌ها یکی از مهمترین گامها در به کارگیری لینوکس و نرم‌افزارهای متن‌باز است، سزاوار است که به بررسی هر یک از این عناصر فهرست هزینه‌ها که متن‌باز بر آنها تأثیر مثبت یا منفی دارد، بپردازیم. در فصول بعدی برخی از این عناصر با جزئیات بیشتری بررسی می‌شود.

سخت‌افزار

نرم‌افزارهای متن‌باز به روشهای گوناگونی بر هزینه‌های سخت‌افزار تاثیر می‌گذارند. در اینجا خصوصیات سیستم‌عامل لینوکس نقش کلیدی بازی می‌کنند. همان طورکه در فصلهای ۱و۲ اشاره شد، به طور نظری همهٔ نرم‌افزارهای متن‌باز شامل لینوکس روی تجهیزات کامپیوتری کالایی باز[1] طراحی شده‌اند و چون در این حالت احتیاجی به وابستگی به رویکردهای سخت‌افزاری اختصاصی نیست، هزینه سخت‌افزار می‌تواند به طور چشمگیری کاهش یابد. در این حالت فرض می‌شود که ترکیب یک سکوی (سخت‌افزاری) باز و لینوکس نیاز به مقیاس‌پذیری، قابلیت اطمینان، در دسترس بودن و کارایی را برآورده می‌کند. نکته قابل توجه دیگر، اکوسیستم سخت‌افزاری (تجهیزات

[1] open commodity computing

سخت‌افزاری که با یکدیگر سازگارند) موجود در نتیجه استفاده از سکوی باز است. این مسئله بدین معنی است که رده گسترده‌ای از افزودنیهای سخت‌افزاری (مؤلفه‌هایی که برای بالا بردن کارآیی به کامپیوتر اضافه می‌شوند) پشتیبانی شده، موجود می‌باشد. این روش، گزینه‌های موجود پیش روی شما را افزایش خواهد داد و تأثیرات رقابتی نیز هزینه‌های شما را کاهش خواهد داد.

بسیاری از کاربران اظهار داشته‌اند که ترکیب لینوکس و نرم‌افزارهای متن‌باز روی پیکربندیهای سخت‌افزاری خیلی قدیمی نیز به خوبی اجرا می‌شود. لینوکس اغلب به عنوان راهی برای استفادهٔ مجدد، روی سخت‌افزارهای قدیمی استفاده می‌شود. همچنین سخت‌افزارهای دارای کارآیی پایین نیز اغلب می‌توانند نیازهای کسب و کار شما را با هزینه‌های خیلی کمتری برآورده کنند. با این همه ممکن است سخت‌افزارهای اختصاصی دست دوم یا بخشی از سیستم موجود برای بهره‌مند شدن از مزایای لینوکس به کار گرفته می‌شود.

سیستم‌عامل

هزینه استفاده از سیستم‌عاملهای اختصاصی روی سیستمهای بزرگ می‌تواند به راحتی به صدها هزار دلار برسد. اما لینوکس را می‌توان بدون هیچ هزینه‌ای (مربوط به مجوز) دریافت، نسخه برداری و نصب کرد. مسلماً مهمترین هزینه مربوط می‌شود به نگهداری، ارتقاء و پشتیبانی سیستم‌عامل در طول چرخه حیات آن است که همراه با پیشرفت کار شکل می‌گیرد. این مسأله را بعداً با تفصیل بیشتری بررسی خواهیم کرد. بدیهی است که انتظار می‌رود ترکیب سخت‌افزار و سیستم‌عامل نیازهای تجاری شما را برآورده کند.

همان گونه که در فصل ۵ توضیح داده شد، شما قادرید مؤلفه‌های مختلف را برای ساخت توزیع لینوکس دلخواه خود ترکیب کنید. به جای تهیه جداگانه مؤلفه‌های مورد نیاز، می‌توانید اجزای متن‌باز دلخواه را با سیستم‌عامل خود همراه کنید.

بسته به نوع کاربرد و اندازه سازمان، تغییر توزیعهای تجاری لینوکس درجهت نیازهای خاص سازمان، می‌تواند هزینه‌های جدیدی به شکل قراردادهای نگهداری و پشتیبانی در پی داشته باشد.

ابزارهای مدیریت و کمکی

تقریبا هر شرکت بزرگی مجموعهٔ گسترده‌ای از ابزارهایی که خاص خودش است ایجاد می‌کند. در اکثر موارد این ابزارها سالها پیش توسعه یافته‌اند و توسعه‌دهنده مربوطه شرکت را ترک کرده است. در برخی موارد کد منبع این ابزارها و سرویسهای داخلی را نیز نمی‌توان پیدا کرد. اینجاست که هزینه‌های نگهداری و انتقال این ابزارهای داخلی به سکوهای دیگر می‌تواند بسیار قابل توجه باشد.

زیبایی متن‌باز اینجا نمایان می‌شود. تقریباً هر عملکردی که فکرش را بکنید توسط توسعه‌دهنده‌ای در جامعه متن‌باز نوشته شده است. اگر هم دقیقاً آنچه می‌خواهید موجود نباشد، چیزی نزدیک به آن یافت می‌شود که می‌توانید آن را متناسب با نیازهای خود تغییر دهید. قسمت جالب این ماجرا این است که وقتی شما ابزاری را توسعه می‌دهید، می‌توانید آن را به جامعه برگردانید تا آن را به طور پیوسته نگهداری کند. بدین ترتیب فرصت عالی برای کاهش هزینه‌ها در اختیار شما قرار می‌گیرد. همکاری با جامعه متن‌باز حسن نیت شما را نشان می‌دهد و در مقابل جامعه را به یاری شما علاقه‌مند می‌نماید.

ارتقاء رویکرد

ارتقاء آنچه که از آن استفاده می‌کنید، همیشه با شرایط و زمان‌بندی دلخواه شما انجام می‌شود. هیچگاه لازم نیست نگران از دست دادن پشتیبانی یک عرضه‌کننده باشید. چون کد منبع همیشه موجود است، هر وقت که بخواهید می‌توانید خودتان اقدام به پشتیبانی محیط موجود در سازمان کنید. وقتی که با پروژه‌های متن‌باز بزرگی چون لینوکس و کارگزار آپاچی سروکار دارید، می‌توانید از پشتیبانی شرکتهای متعددی که در این زمینه وجود دارد، بهره‌گیرید یا می‌توانید با حمایت جامعه متن‌باز خودتان این پشتیبانی را برعهده گیرید.

فاکتور دیگر در هزینه‌های ارتقاء، به مجوزهای ارتقاء مربوط می‌شود. بسیاری از عرضه‌کنندگان نرم‌افزار برای انجام به‌روزرسانیهای کوچک هزینه‌های نگهداری را دریافت می‌کنند و در مورد ارتقاء به نسخه‌های جدید هزینه‌های سنگینی مطالبه می‌کنند.

مدیریت و راهبری و ردیابی مجوزها

وقتی که در فصل هشتم مقوله به کارگیری را با تفصیل بیشتری مورد بررسی قرار دهیم، خواهید فهمید که یکی از مواردی که در به کارگیری یک محیط لینوکسی باید لحاظ شود، توانایی جای دادن آن در زیرساخت موجود است. شرکتها سالانه میلیونها دلار فقط صرف ردیابی مجوزهای نرم‌افزار خود می‌کنند. بسیاری از آنها ابزارهای پیچیده‌ای برای ردیابی مجوزهای خود در سطح جهانی ایجاد می‌کنند، در حالی که شرکتهایی هم وجود دارند که برای توافق با فروشندگان در زمینه مجوزهای شرکتی و به حداقل رساندن هزینه‌های ردیابی مجوز حاضر به انجام هر کاری هستند. در مورد نرم‌افزارهای متن‌بازی که تحت مجوزهای قابل قبول برای OSI توزیع شده‌اند، نیازی به ردیابی مجوزها نیست و اگر هر تعداد نسخه از نرم‌افزار را که نیاز دارید نسخه برداری کنید کاملاً قانونی است.

مجوزهای برنامه‌های کاربردی و دیگر نرم‌افزارها

واضح است که یکی از بهترین جاهایی که می‌توانید هزینه‌ها را کاهش دهید، هزینه مجوز نرم‌افزار است. برای بسی‌اری از افراد این یک مفهوم جدید است. در اکثر موارد شما واقعاً نرم‌افزار متن‌باز را خریداری می‌کنید. نمونهٔ آن خرید یک توزیع لینوکس بصورت بسته‌بندی شده از یک فروشگاه است. ولی تفاوت عمده اینجاست که شما هزینه‌ای بابت مجوز آن نمی‌دهید. در حقیقت شما پول راحتی بدست آوردن نرم‌افزار (بجای دریافت آن از اینترنت) می‌دهید. زیرا در این صورت این نرم‌افزار در شکلهای مختلف رسانه (DVD,CD برای بارکردن نرم‌افزار) به همراه مستندات و احتمالا پشتیبانی محدود در اختیار شما قرار می‌گیرد.

وقتی مقوله پشتیبانی را با تفصیل بیشتری بررسی کنیم با گزینه‌های موجود آشنا خواهید شد. در اینجا کافی است آگاه باشید که نیازی به پرداخت هزینه مجوز برای نرم‌افزارهای متن‌باز نیست. شما می‌توانید نرم‌افزار را از هر جایی دریافت کنید، سی‌دی‌های دلخواه خود را بسازید و حتی از مستندات را روی کاغذ چاپ کنید.

هزینه‌های پشتیبانی

یکی از متداولترین مدلهای تجاری مرتبط با متن‌باز، ارائه پشتیبانی نرم‌افزارهای متن‌باز است. وقتی با نرم‌افزارهای اختصاصی مواجه می‌شوید، گزینه‌های پشتیبانی به طور قابل توجهی محدود می‌شود و تنها می‌توانید از عرضه‌کننده‌ای که نرم‌افزار را فروخته است، پشتیبانی بگیرید. در این حالت عرضه کننده، مشتری را سرگردنه گیرآورده و برای خدمات پشتیبانی خود تقاضای اجرت سنگینی می‌کند. حتی اگر بنا دارید با نرم‌افزارهای متن‌باز به طریقی مشابه با نرم‌افزارهای تجاری برخورد کنید، این واقعیت ساده که خدمات پشتیبانی از طرف چندین عرضه‌کننده موجود است، رقابت را افزایش و تاثیر مثبتی روی کاهش قیمت دارد. عملاً شما نیاز به تغییر فرآیندهای تجاری خود برای در دست داشتن پشتیبانی به عنوان یک گزینه رقابتی دارید.

خرابیهای پیش‌بینی‌نشده

در دنیای نرم‌افزارهای اختصاصی، برخی نقاط ضعف می‌تواند سبب خرابیهای نامطلوب و غیرمنتظره شود. در مورد محیطهای دارای ماموریتها و وظایف بحرانی، اغلب سرمایه‌گذاریهای هنگفتی برای به حداقل رساندن این موارد می‌شود. به هر حال شما در کنترل عرضه‌کننده هستید. اگر عرضه کننده تمایلی به اعمال تغییرات مورد نظر شما در بازهٔ زمانی دلخواه شما نداشته باشد، معمولاً کاری از دستتان برنمی‌آید.

مجدداً حتی در شرایطی که شما با نرم‌افزارهای متن‌باز همانند نرم‌افزارهای تجاری کار می‌کنید، امکان استفاده آزادانه از جامعه متن‌باز یا در اختیار داشتن کادری از کارشناسان فناوری اطلاعات داخل سازمانی برای حل مشکلات، می‌تواند هزینه خرابیهای پیش‌بینی‌نشده را کاهش دهد.

آموزش

همانطور که در فصل ۱ اشاره شد، وجود منابع انسانی آموزش دیده، یکی از مزایای کلیدی حرکت رو به جلوی لینوکس است. اگر شما یک استاندارد صنعتی یا یک محیط متن‌باز را گسترش می‌دهید، هزینه‌های آموزش با توجه به این واقعیت که منابع انسانی آموزش دیده فراوانی وجود دارند، کاهش می‌یابد. با این حال اگر در حال مهاجرت از محیط موجود هستید، هزینه‌های آموزش نیاز به سرمایه‌گذاری عظیمی دارند که باید لحاظ شوند.

صرف نظر از نوع استفاده، نرم‌افزارهای متن‌باز، رفتارهای متفاوتی از جانب توسعه‌دهندگان، مذاکره کنندگان قراردادها، کادر خرید و تیم مدیریت می‌طلبند. این کتاب جزئیات مورد نیاز برای درک تفاوت این رفتارها را شرح خواهد داد، ولی خودتان باید هزینه‌های مربوطه را لحاظ کنید.

پذیرش یک رویکرد ناقص

در روزهای آغازین عصر کامپیوتر، اکثر محیطها (راه‌حلها) از پایه ساخته می‌شدند. در نتیجه نیازهای سازمان را بخوبی برآورده می‌کردند. اما توسعه این راه‌حلها کار بسیار هزینه‌بری بود. به مرور زمان عرضه‌کنندگان نرم‌افزار، برنامه‌هایی کاربردی ساختند که مسائل و نیازهای زیادی را پوشش می‌دادند. اگرچه این راه حل‌ها کامل نبودند، اما هزینهٔ تهیهٔ آنها به طور چشمگیری کمتر بود، به طوری که انتخاب یک راه حل ناقص ترجیح داده می‌شد. به هر حال این هزینه‌ای است که باید در کسب و کار خود لحاظ کنید و یکی از مهمترین مزایای متن‌باز این است که این هزینه‌ها را در بر ندارد.

اینجا نقطهٔ درخشش متن‌باز است. توانایی دراختیار گرفتن نرم‌افزار متن‌بازی که بتواند نیازهای تجاری شما را تقریب بزند و سپس بتوانید آنرا با نیازهای خاص خود تطبی‌ق‌دهنده، بهترین گزینه ممکن است. همانطور که قبلاً اشاره شد در حالات زیادی جامعهٔ متن‌باز از پیشرفتهای شما استقبال خواهد کرد؛ آنها را خواهد پذیرفت و کل محیط را پشتیبانی خواهد نمود.

به حساب آوردن این مزایا می‌تواند مشکل باشد، چرا که شما باید زمینه‌ای را تحلیل و مقایسه کنید که هزینه آن صفر است علاوه بر اینکه هزینه‌های افزاینده را نیز باید لحاظ کنید. با این تحلیل می‌توانید یک مورد تجاری بسازید. جدول ۶-۱ راهنماییهایی برای این تحلیل ارائه می‌دهد:

جدول ۶-۱ پیاده سازی یک راه حل "کامل"

هزینه‌های کاهش یافته یا حذف شده	هزینه‌های جدید یا افزایش یافته	هزینه صفر

هزینه مجوز نرم‌افزارها در راه حل "ناکامل"	به کارگیری راه‌حلهای متن‌باز برای نیازهای تجاری	پذیرفتن راه حل نرم‌افزار متن‌باز
پرسنل اضافی در راه‌حل ناکامل	نگهداری و پشتیبانی داخلی از هر جزئی که جامعه متن‌باز آن را نپذیرد	پذیرفتن کارهایی که جامعه متن‌باز برروی نرم‌افزار انجام داده است
راه حل‌های موقتی	پرسنل برای تأثیرگذاری و حفظ ارتباط با جامعه متن‌باز	بروز رسانی پیوسته توسط جامعه متن‌باز
سفارشی کردن	پرسنل حرفه‌ای	شروع کردن از کدهای منتشر شده موجود
تأثیرات تجاری دیگر (مانند زمان عرضه در بازار، رضایت مشتری و غیره)		کنترل کیفیت گسترده توسط جامعه متن‌باز

بدیهی است که این جدول مقایسات هزینه، باید با دیگر فاکتورهای هزینه - که در همین فصل شرح داده شده‌اند - ترکیب شود. نکته قابل توجه این است که متن‌باز تلاش می‌کند که فرصتی برای رسیدن به یک رویکرد تقریباً کامل با معقولترین هزینه، فراهم کند. تحلیلهای تجاری شما تعیین خواهد کرد که آیا رویکرد جدید ارزش سرمایه‌گذاری دارد یا خیر.

فراهم کردن لینوکس و نرم‌افزارهای متن‌باز

شرکت شما، معمولاً هنگام در اختیارگرفتن محصولات نرم‌افزاری سنتی فرایند جامع و فراگیری برای ایجاد ارتباط با عرضه‌کنندگان و تولیدکنندگان راه‌حلها ایجاد می‌کند. شما با توجه به حجم نرم‌افزارها در مورد هزینه‌های مجوز آنها مذاکره می‌کنید و تعداد محدودی یا حتی یک تأمین‌کننده را برای این کار انتخاب می‌کنید تا هزینه‌ها را به حداقل برسانید. فرآیند دراختیارگرفتن لینوکس، آپاچی و یا نرم‌افزارهای متن‌باز دیگر کلاً با یکدیگر متفاوت است. مثالی از در اختیار گرفتن سیستم‌عامل لینوکس را در نظر بگیرید. موارد زیر معمولترین راه‌های داشتن یک نسخه از لینوکس هستند:

- خرید یک جعبه حاوی رسانه نصب کننده لینوکس از یک فروشنده محلی

- دریافت تصاویر لوحهای فشرده آن از اینترنت
- استفاده از لوح فشرده‌ای که به همراه یک کتاب یا محصول عرضه می‌شود
- استفاده از نرم‌افزاری که در یک سیستم کامپیوتری نصب شده یا همراه تجهیزات جانبی کامپیوتر است

روشهای مشابهی برای در اختیار گرفتن نرم‌افزارهای متن‌باز دیگر وجود دارد. در برخی موارد عرضه‌کنندگان نرم‌افزار، برنامه‌های کمکی متن‌باز، ابزارها و کاربردهای اضافی را با نرم‌افزار خود همراه می‌کنند.

عاملان خرید در سازمان شما، اغلب برای در اختیار گرفتن و داشت حساب مجوز نرم‌افزارهایی که خریداری می‌شوند، آموزش داده می‌شوند. این احتمال وجود دارد که آنها بخواهند همین فرآیندها را روی نرم‌افزارهای متن‌باز اعمال کنند. به هر حال مهم است که مأموران خرید شما به گونه‌ای آموزش دیده باشد که تنها نرم‌افزارهای متن‌بازی را خریداری کند که ارزش صرف هزینه را دارند. در مواردی که عرضه‌کنندگان سخت‌افزار یا نرم‌افزار، نرم‌افزارهای متن‌باز نشان‌دار را با محصول خود همراه می‌کنند، دقت کنید که ممکن است این عرضه کننده هزینه مجوز این نرم‌افزار را روی قیمت محصول خود کشیده باشد. همراه سازی نرم‌افزارهای متن‌باز نشان دار با یک محصول به قرادادهای تجاری مربوط می‌گردد که بین دو شرکت بسته می‌شود که اجازه این کار را به آنها می‌دهد.

در اکثر موارد عرضه‌کنندگان سخت‌افزار، نرم‌افزار، مؤلفه‌های متن‌بازی را همراه محصول خود می‌کنند که در جای دیگر بدون هیچ هزینه‌ای موجود باشند. به هر حال ممکن است در مواردی تسهیلاتی که همراه سازی فراهم می‌کند، ارزش پرداخت هزینهٔ اضافه را داشته باشد. نکته اینجاست که باید متوجه باشید به ازای پولی که پرداخت می‌کنید، چه دریافت می‌کنید. امکان بدست آوردن نرم‌افزارهای متن‌باز از منابع دیگر یا نسخه‌برداری به تعداد دلخواه همیشه ممکن است. در اینجا نیازی به ردیابی مجوز نرم‌افزارها نیست.

قراردادها

اگر شما عرضه کننده سخت‌افزار یا نرم‌افزار هستید، طبیعی است که از عرضه‌کننده‌ای که محصول وی را با محصول خود همراه می‌کنید، در مقابل صحت محصول وی مطالبه ضمانت نمائید. اما وقتی حرف از نرم‌افزارهای متن‌باز به میان می‌آید، قواعد تغییر می‌کند. از آنجایی که نرم‌افزارهای متن‌باز توسط مجموعه‌ای از افراد مختلف نوشته می‌شوند و بدون اغراق صدها مالک حق انحصاری مختلف برای آنها وجود دارد، کسی برای شما چنین تضمینی نمی‌کند.

به خاطر داشته باشید در هر مذاکره‌ای با یک عرضه کننده تجاری، شما تنها در مورد هزینه مجوز نرم‌افزار مذاکره نکرده بلکه اغلب برای پشتیبانی، خدمات و مواردی از این قبیل مذاکره می‌کنید. همچنین به خاطر داشته باشید که این محیط رقابتی است و گزینه‌های زیادی برای انتخاب وجود دارد.

تغییردادن نرم‌افزار متن‌باز

بسیاری از تحلیل‌گران صنعتی، تغییر نرم‌افزار متن‌باز را توصیه نمی‌کنند و پیشنهاد دارند که با این قبیل نرم‌افزارها همانند دیگر نرم‌افزارهای اختصاصی در شرکت رفتار شود. این توصیه از این واقعیت نشأت می‌گیرد که با تغییر دادن نرم‌افزار طبق نیازها و استفاده‌های خاص، باید از پشتیبانی که برای نرم‌افزار جدید انتظار می‌رود، چشم پوشی کرد. شاید برای شرکتهای کوچکتر با پرسنل کمتر در فناوری اطلاعات، این دلیل قانع کننده‌ای برای تغییر ندادن نرم‌افزارهای متن‌باز باشد، ولی برای شرکتهای بزرگ که با جامعهٔ متن‌باز تعامل دارند، تغییر نرم‌افزار می‌تواند کمک خوبی برای کاهش هزینه‌ها باشد.

تغییر یک نرم‌افزار متن باز بهتر است همیشه با هماهنگی نگهدارندگان آن در جامعه متن‌باز انجام شود. توانایی تغییر نرم‌افزار مزایایی را هم برای شرکتهای بزرگ و هم برای تولیدکنندگان نرم‌افزار فراهم می‌کند:

- تأثیر در جهت پیشرفت محصول نرم‌افزاری

- سرمایه گذاری در عملی کردن اولویتها با زمانبندی مطلوب
- داشتن دید عمیق و کلی از داخل سازمان برای طرح ریزیهای آینده

توجه کنید که نگهدارنده کنترل نرم‌افزار را در دست دارد. مشارکت پیوسته باعث تأثیرگذاری بر وی می‌شود. هنگام تحلیل هزینه‌های لینوکس و متن‌باز، مقایسه هزینهٔ انتظار برای عملی شدن اولویتها را با هزینه‌های سرمایه‌گذاری و مشارکت مستقیم خودتان در احقاق اولویتها را نیز در نظر بگیرید.

اگر مایل نیستید که به انجمن بپیوندید و یا قادر نیستید که این کار را انجام دهید و نمی‌توانید در آینده هر محصول نرم‌افزاری شرکت کنید، آن وقت است که باید حق را به تحلیل گران بدهید؛ بهتر است که از هر تغییر در سیستم متن‌باز بپرهیزید و فقط زمانی از آن استفاده کنید که بسته‌بندی شده و توسط یک عرضه‌کننده تجاری حمایت می‌شود.

جمع بندی

شما اکنون باید متوجه شده باشید که لینوکس و متن‌باز چگونه هزینه‌های یک راه حل در سازمان را تحت تاثیر قرار می‌دهند. در اکثر موارد می‌توانید از مزایای عمدهٔ آن بهره برید و هزینه‌ها را کاهش دهید. به هر حال شما متوجه این مسأله هم شدید که به کارگیری متن‌باز به این معنی نیست که هزینه‌ها به صفر می‌رسد و متحمل هیچ هزینه‌ای نخواهید بود. برای نرم‌افزارهای متن‌باز هیچ حق الامتیازی مربوط به مجوزها نخواهیم داشت و نیاز نیست برای آنها مبلغی پرداخت شود. حتی اگر با مؤلفه‌های دیگری همراه شوند. ولی در عین حال برای نگهداری راه حل خود باید هزینه پرداخت کنید. فرآیندهای مذاکره برای قرارداد ممکن است نیاز به تجدید نظر داشته باشند. نهایتاً اینکه شما می‌توانید کنترل بیشتری روی هزینه‌ها از طریق تصمیم‌گیریهای معقولانه و روشن در مورد تغییر دادن نرم‌افزار و برگرداندن آن به جامعه و بهره‌گیری از مزایای رویکردی که بیشترین نیازهای تجاری شما را برآورده می‌کند، داشته باشید.

یکی از عناصر اصلی در پائین نگهداشتن هزینه‌ها در اکوسیستم، استانداردها هستند. فصل بعدی استانداردهای خاص محیط لینوکس را بررسی خواهد کرد. ارتقاء این استانداردها، رشد اکوسیستم را تسریع نموده و هزینه‌ها را کاهش خواهد داد.

فصل ۷

استانداردها -یک لینوکس

اکنون شما اتوموبیل خود را تکمیل کرده‌اید. این اتوموبیل را با ساختن موتور آن آغاز کردید. سپس امکانات مورد نیاز را به آن افزودید. اکنون از هزینه‌های آتی اتوموبیل خود نیز آگاهید. حال تصور کنید اتوموبیل افراد مختلف آنقدر متفاوت باشند که نتوانند در یک جاده حرکت کنند و از یک سوخت استفاده کنند. در این صورت استفاده از اتوموبیل در انحصار افرادی خواهد بود که توانایی ایجاد جاده‌ها و بزرگراه‌های اتوموبیل خود را داشته باشند. در یک اتوموبیل، داشتن امکانات متناسب با نیازهای مشتریان به اندازه ملزم بودن به رعایت استانداردها مهم است.

در این فصل جزییات استانداردهای تعیین شده در حوزه لینوکس و متن‌باز را بررسی خواهیم کرد. در نظر داشته باشید که فقط استانداردهای جدید و مخصوص لینوکس مورد بحث قرار خواهند گرفت. همچنین جنبه‌های مختلف یک استاندارد را که باعث قابل انتقال بودن نرم‌افزار می‌شوند، بررسی خواهیم کرد. به همین دلیل اهداف این فصل درک موارد زیر است:

- چرا به استانداردهای لینوکس و متن‌باز احتیاج داریم
- استانداردهای کلیدی و نحوهٔ کار آنها
- تطابق و آزمایش کردن
- جایی که از استانداردها استفاده نمی‌شود

اینها اطلاعات لازم برای استفاده مؤثر از مجموعه استانداردهای متن‌باز را در اختیار شما قرار می‌دهند. بدین ترتیب می‌توانید از تمامی مزایای یک محیط باز استفاده کنید.

چرا استاندارد؟

همان گونه که در فصل قبل فرا گرفتید، توزیع لینوکس ترکیبی است از کرنل لینوکس، پیمانه¬ها، وصله‌ها، ابزارهای GNU و مجموعه‌ای از برنامه‌های دیگر. کار فروشندگان توزیعهای لینوکس را در زمینهٔ بسته‌بندی لینوکس باید تقسیم کرد. اکنون نصب و استفاده از لینوکس امری بسیار آسان شده است. اما این روح همکاری و نیاز به نوآوری باعث ایجاد تفاوتهای زیادی میان توزیعهای مختلف شده است. از این تفاوتها هیچ کس سودی نمی‌برد. در اینجا به چند نمونه از تصمیماتی که فروشندگان توزیعها باید بگیرند، اشاره می‌شود:

- نسخهٔ کرنل
- وصله‌ها
- ماجولها، درایورها و نسخه‌ها
- موقعیت فایلها
- نسخه‌های بسته‌های نرم‌افزاری

تقریباً همه افراد درگیر در صنعت لینوکس (کاربران، فروشندگان توزیعها و فروشندگان برنامه‌های کاربردی) به این نتیجه رسیده‌اند که این تفاوتها باعث تأخیر در به کارگیری لینوکس می‌شوند. در نتیجه فروشندگان بزرگ لینوکس به همراه فروشندگان سخت‌افزار و نرم‌افزار و برنامه‌نویسان سرشناس متن‌باز با یکدیگر متحد شدند تا مجموعه استانداردهایی برای لینوکس تعریف کنند. این استانداردها توسط گروه استانداردهای آزاد [1] مدیریت می‌شوند. مسلماً باید مزایایی در تعیین این استانداردها وجود داشته باشد. قبل از اینکه به جزئیات این استانداردها نگاهی بیاندازیم، مزایای عمدهٔ این استانداردها برای افراد دخیل در این صنعت را برمی‌شماریم.

- **کاربران-** کاربران نهایی اغلب شرکتهای متوسط را بزرگ با حضور جهانی هستند. استانداردهای لینوکس به کاربران این امکان را می‌دهد که نیازهای محلی خود را در

[1] Free Standards Group

انتخاب توزیع لینوکس دخیل کنند و در عین حال از سازگاری آن با نرم‌افزارهای استاندارد مطمئن باشند. مزیت دیگر استانداردها برای کاربران این است که حداکثر نرم‌افزارهای موجود را در توزیعهای منتخب خود استفاده کنند.

- **فروشندگان توزیعها-** سرمایه لازم برای ایجاد یک توزیع لینوکس کم نیست. هر چه استانداردها بیشتر باشد، تشابه بین توزیعها بیشتر می‌شود و در نتیجه هزینه کمتری دربر خواهد داشت. همچنین استانداردها باعث بزرگ شدن بازار می‌شود. اینترنت را در نظر بگیرید. اگر استانداردها در این زمینه وجود نداشتند، اندازه اینترنت بسیار کوچکتر می‌بود و ما شاهد تغییرات صنعتی اواخر دهۀ ۱۹۹۰ نمی‌بودیم. فروشندگان توزیعها از ۲ جنبه سود می‌برند: ۱) حفظ سازگاری بین نسخه‌های مختلف؛ ۲) داشتن تعریف مشخصی از محل اتمام لایۀ اشتراک و آغاز لایۀ منحصر به فرد ارزش افزوده.
- **فروشندگان نرم‌افزار-** تجربۀ تولید نرم‌افزار برای گونه‌های مختلف یونیکس اثبات کرده است که این امر بسیار پرهزینه است. فروشندگان نرم‌افزار مجبور هستند که بین هزینه پشتیبانی نسخه‌های مختلف و بزرگی بازار، تعادل برقرار کنند. استانداردها این امکان را می‌دهد که فروشندگان نرم‌افزار یک نسخه از هر نرم‌افزار را تولید کنند و آنرا تحت بسترهای مختلف پشتیبانی کنند.

لینوکس و نرم‌افزارهای متن‌باز امروزه در حد خوبی رشد نموده‌اند و امروزه برای اطمینان از حفظ رشد این صنعت نیازمند داشتن استانداردهایی هستیم که بر نوآوری متمرکز باشند.

گروه استانداردهای آزاد

گروه استانداردهای آزاد (www.freestandards.org) سازمانی غیرانتفاعی است که اولین وظیفۀ آن گسترش استفاده و مرغوبیت تکنولوژیهای متن‌باز، از طریق استانداردها، می‌باشد. محصول طبیعی

کار آنها این خواهد بود که لینوکس به سمت مرحلهٔ بعدی پشت بازار یعنی بستر شرکتی برای برنامه‌های تجاری حرکت خواهد کرد. گروه‌های کاری اصلی این سازمان به شرح زیر می‌باشند:

- **پایه استانداردهای لینوکس (LSB)**- LSB مجموعه‌ای از استانداردها و آزمایشها برای رابطها می‌باشد. این استانداردها باعث قابل انتقال بودن برنامه‌ها از توزیعی به توزیعی دیگر صرفنظر از نوع و نسخه آن می‌شوند.
- **طرح بین‌المللی‌سازی لینوکس (Li18ux)**- این استاندارد بستری برای بومی‌سازی زبانی و فرهنگی، توزیعها و برنامه‌ها فراهم می‌کند.
- **استاندارد سلسله مراتب فایل سیستم (FSH)**- این استاندارد مدل ثابتی برای محل قرار دادن فایلها در یونیکس و امروزه لینوکس، تبلیغ می‌کند. این یکی از استانداردهای کلیدی است که LSB به آن وابسته است. استاندارد FSH به همراه LSB بررسی خواهد شد.

گروه استانداردهای آزاد همهٔ استانداردها را به تنهایی تعیین نمی‌کند، بلکه با طراح استاندارد دیگری برای لینوکس نیز همکاری دارد: گروه X Desktop. گروه X Desktop بر روی استانداردهایی فعالیت می‌کند که همکاری و قابلیت جابجایی بین محیطهای رومیزی لینوکس و یونیکس را تشریح می‌کند. اکنون می‌دانید که دو محیط رومیزی اصلی برای لینوکس وجود دارد: GNOME و KDE. گروه استانداردهای آزاد همکاری نزدیکی نیز با گروه XFree86 (به فصل ۴ مراجعه کنید) دارد.

گروه غیرانتفاعی استانداردهای آزاد توسط شرکتهای بزرگ این صنعت تأسیس شده است. این شرکتها به خوبی مزایای وجود داشتن این استانداردها را می‌دانند. هر کاربر یا تولیدکنندهٔ نرم‌افزار و سخت‌افزاری تشویق به همکاری در ایجاد این استانداردها می‌شود. استانداردها وقتی موفق هستند که نیازهای کاربران را برآورده کنند و به طور گسترده‌ای در جامعه به کار گرفته شوند.

پایه استانداردهای لینوکس (LSB)

LSB (www.linuxbase.org) مهمترین استانداردی است که باعث ازدواج موفق توزیعهای لینوکس و برنامه‌های لینوکس می‌شود. این استاندارد از اجزاء زیر تشکیل شده است:

- مشخصات دودویی رابط (مشخصات دودویی الزام می‌کند که برنامه‌ها نباید موقع انتقال از یک سیستم به یک سیستم دیگر نیازمند کامپایل مجدد باشند). که باعث سازگاری نرم‌افزارها با تمامی توزیعهای لینوکس تهیه شده می‌شود.
- مجموعه‌ای از آزمایشها بر روی توزیعها و برنامه‌ها برای اطمینان از تطابق با LSB.
- ایجاد محیطی برای گسترش نرم‌افزارهای سازگار با LSB.
- نمونه‌های پیاده شده LSB برای اجرا و آزمایش برنامه‌ها در یک محیط LSB خالص.

با به کارگیری استانداردها و ابزارها فروشندگان می‌توانند توزیعهای لینوکس منطبق با LSB و فروشندگان نرم‌افزارهای منطبق با LSB تولید کنند.

تولیدکنندگان نرم‌افزار با تمرکز بر مجموعه رابطهای مشترک می‌توانند برنامهٔ خود را مطابق LSB تولید و آزمایش کنند و مطمئن باشند که این برنامه بر روی هر توزیع منطبق با LSB اجرا می‌شود. شرکتهایی هم که تیم برنامه‌نویسی دارند باید همهٔ برنامه‌های خود را مطابق با LSB بنویسند. بنابراین قبل از تهیهٔ توزیع لینوکس یا هر برنامه‌ای از تطابق آن با آخرین نسخهٔ LSB اطمینان حاصل کنید. بدین طریق سرمایه‌گذاری شما ماندگاری بیشتری خواهد داشت.

به LSB می‌توان به عنوان جمع‌آوری کنندهٔ مجموعه استانداردهای در حال استفاده در صنعت و مورد تأیید جامعه برنامه‌نویسان لینوکس، نگاه کرد. یکی از اهداف مشخص LSB این است که استانداردهای موجود را مستندسازی کند نه اینکه استاندارد جدیدی اختراع کند. LSB اختراع مجدد نیست بلکه گسترش چیزی است که قبلاً اختراع شده است. به همین دلیل این استاندارد به راحتی مورد قبول تولیدکنندگان برنامه کاربردی و توزیعهای لینوکس قرار گرفته است و تطابق آنها با این استاندارد مشکل نیست.

از آنجایی که LSB یک مشخصات دودویی است، لازم است که به دو جزء اصلی تقسیم شود: عمومی و بسته به نوع پردازنده؛ عمومی برای چیزهایی که مستقل از دستگاه هستند و بسته به نوع پردازنده فقط برای چیزهایی که باید وابسته به نوع دستگاه باشد. شکل ۷-۱ نشان می‌دهد که چگونه با تلفیق این دو بخش می‌توان یک محیط منطبق با LSB برای یک معماری خاص طراحی کرد.

در اینجا قصد نداریم کل استاندارد LSB را بازخوانی کنیم. یک نسخه از این استاندارد را می‌توانید از سایت LSB (www.linuxbase.org/spec) دریافت کنید. خالی از لطف نخواهد بود که نگاهی سطح بالا بر این استاندارد و نحوهٔ کنار هم آمدن اجزاء مختلف بیاندازیم.

اجزای توزیعهای LSB

LSB مجموعه رابطهایی را که برنامه‌ها باید داشته باشند تعریف می‌کند. شکل ۷-۲ اجزاء استاندارد شدهٔ یک توزیع لینوکس منطبق با LSB را نشان می‌دهد.

شکل ۷-۱: پشتیبانی LSB از معماریهای پردازنده‌ها

شکل ۷-۲: توزیع مطابق با LSB

در جدول ۷-۱ اجزای مهم LSB عمومی (gLSB) در یک توزیع لینوکس و تأثیر آنها بر برنامه‌ها را شرح می‌دهد:

جدول ۷-۱ اجزای gLSB

جزء LSB	شرح
رابط کرنل لینوکس	گروههای توسعه کرنل باید در نوآوریها و گسترش تواناییهای کرنل آزاد باشند. LSB ساخته شد تا نوآوری را گسترش دهد. با استفاده از رابط POSIX (استاندارد بخوبی جاافتاده در زمینه رابطهای یونیکس) در عین تشویق نوآوری، سازگاری برنامه‌ها را تضمین می‌کند. مفهوم این کار این است که اجازه داده می‌شود که ساختار کنونی یونیکس نیز با LSB تطابق کند.
چارچوب بسته بندی	بسته بندی به فرآیند ترکیب مجموعه فایلهای تشکیل دهنده یک برنامه، برای استفاده آسان‌تر، گفته می‌شود. با تعیین چارچوب بسته بندی، فروشنده‌ها می‌توانند از مجموعه ابزار موجود برای پشتیبانی این چارچوب انتخاب کنند.
پیوندسازی پویا	پیوندسازی پویا قالب دودویی و فیلدهای مورد نیاز برنامه را مشخص می‌کند. پیوندسازی پویا این امکان را فراهم می‌آورد که برنامه‌های زیادی بتوانند از کتابخانه‌های توابع مشترکی استفاده کنند.
محیط اجرا	محیط اجرا بر استاندارد FSH و امنیت اجرا تمرکز یافته است. همانگونه که قبلاً اشاره شد، استاندارد FSH از اجزای اصلی LSB می‌باشد. با اطمینان از هماهنگی

جزء LSB	شرح
	موقعیت قرارگیری فایلها در سیستم قابلیت پیش‌بینی برنامه‌ها را افزایش می‌دهد.
راه اندازی سیستم	راه اندازی سیستم وابستگی اکثر برنامه‌ها به یک وضعیت شناخته شده در هنگام راه اندازی سیستم را به رسمیت می‌شناسد. این قسمت از استاندارد، اطمینان حاصل می‌کند که برنامه‌ها محیطی در اختیار دارند که می‌توانند روی آن حساب کنند، مانند اسکریپت‌ها و پیکربندی‌ها.
پوسته استاندارد	در طول سالها تعداد زیادی پوسته (مفسری که دستورات تایپ شده در خط فرمان را اجرا می‌کند) طراحی شده‌اند. این قسمت از LSB استاندارد POSIX را به رسمیت می‌شناسد و در عین حال واقعیتهای پراستفاده‌ترین پوسته در لینوکس را نیز برمی‌شمارد.
قالبهای فایلهای اشیاء	وقتی که مترجمی (کامپایلر) برنامه اجرایی را با استفاده از یک کد منبع می‌سازد، آن فایل را باید براساس قاعده کاملاً مشخصی روی دیسک بنویسد. اگر استاندارد مشخصی برای این قاعده وجود داشته باشد، تمامی برنامه‌ها می‌توانند در حافظه بارگذاری شده اجرا شوند.
کتابخانه‌ها	در نمودار ملاحظه می‌کنید که سه نوع کتابخانه وجود دارد: پایه، گرافیک و کمکی. در تئوری تمامی برنامه‌های بارگذاری شده در سیستم حداقل به یکی از کتابخانه‌های هریک از این سه دسته وابسته است. مشخص کردن اینکه کدامیک از این کتابخانه‌ها باید در دسترس برنامه‌ها باشند باعث می‌شود برنامه‌ها با اطمینان بیشتری اجرا شوند.
دستورات و برنامه‌های کمکی	تمامی سیستمهای یونیکس و لینوکس صدها دستور در اختیار مدیران سیستم و کاربران قرار می‌دهند. این بخش از استاندارد دستورهایی را که باید در اختیار کاربران باشد تصریح می‌کند. به این ترتیب برنامه‌هایی نیز که وابسته به اسکریپتها هستند با اطمینان بیشتری اجرا می‌شوند.
کاربران و گروه‌ها	بازهم با به کارگیری اهرم استاندارد POSIX نام کاربرها و گروه‌ها نیز از استانداردها تبعیت می‌کنند.

همانگونه که گفته شد LSB اجزاء وابسته به معماری را نیز مشخص می‌کند. همانگونه که در فصل دوم اشاره کردم، از آنجایی که اکثر برنامه‌ها تحت معماری Intel IA-32 نوشته می‌شوند، همهٔ طراحیهای اولیه از اینجا آغاز می‌شود. LSB هم مستثنی نیست. در زمان نوشتن این کتاب بخش وابسته به پردازنده LSB در زمینهٔ IA-32 تعریف شده است و پیش‌نویسهای IA-64 در حال تهیه هستند. در جدول ۷-۲ اجزاء LSB برای IA-32 آورده شده است.

جدول ۷-۲ اجزای وابسته به معماری LSB

اجزای LSB	شرح
اطلاعات سطح پایین سیستمی	در این بخش رابط بسیار سطح پایین معماری IA-32 شرح داده شده است. این بخش مطمئن می‌شود که کامپایلرها، استانداردهای مورد نیاز را برای ساختن برنامه‌هایی که بخوبی اجرا می‌شوند دارند.
قالب فایل هدف	همانند مشخصات gLSB در این زمینه، مطمئن می‌شود که کامپایلرها فایلهای هدف را به درستی می‌نویسند.
بارگذاری برنامه‌ها و پیوندسازی پویا	فرآیند اجرای برنامه شامل دو بخش بارگذاری برنامه در حافظه و زمانبندی (پردازنده) برای اجرا آن تشکیل شده است. همانگونه که در gLSB ذکر شده است، در فاز بارگذاری، برنامه باید با تمامی کتابخانه‌های مورد نیاز پیوند برقرار کند. این بخش از استاندارد مطمئن می‌شود که فرآیندهای بارگذاری، پیوندسازی، زمانبندی و جادادن یک برنامه اجرایی در حافظه از قاعده درستی پیروی می‌کنند.
کتابخانه‌های پایه	gLSB مجموعه‌ای از کتابخانه‌های پایه را معرفی می‌کند. تعدادی کتابخانه هم مخصوص معماری IA-32 وجود دارد که در اینجا معرفی می‌شوند.

LSB استانداردی فراگیر و ابزاری است که انتظارات یک برنامه از یک سیستم منطبق با آنرا شرح می‌دهد. همچنین قواعدی وجود دارد که نرم‌افزارها برای تطابق با LSB باید آنها را رعایت کنند.

برنامه‌های منطبق بر LSB

اطمینان از قابل اجرا بودن نرم‌افزارهای در حال طراحی، تحت همهٔ لینوکسها، چه این برنامه برای فروش باشد، چه برای استفاده داخل سازمان برد بزرگی است. با این کار شما این امکان را در اختیار کاربران قرار می‌دهید که نوع لینوکس خود را از مجموعه لینوکسهای موجود انتخاب کنند، نه اینکه مجبور به استفاده از توزیع خاصی باشند. اگر کاربری به هر دلیل مجبور به تغییر توزیع لینوکس خود شود، سرمایه‌ای را که صرف تهیهٔ نرم‌افزار شما کرده است از دست نمی‌دهد.

ساده‌ترین تصور از انطباق نرم‌افزار با LSB این است که می‌تواند بر روی هر توزیع منطبق با LSB اجرا شود. چنین نرم‌افزاری نباید به بخشهای اضافه برخی توزیعها که در LSB نیامده‌اند، وابسته باشد. همچنین اگر چنین برنامه‌ای لازم است با برنامهٔ دیگری ترکیب شود یا وابستگی به آن داشته باشد، آن برنامه نیز باید منطبق بر LSB باشد.

البته برنامهٔ شما باید از تمامی قواعد ذکر شده در LSB پیروی کند. در ادامه آزمایش تطابق را بررسی خواهیم کرد که به شما اطمینان می‌دهد نرم‌افزارتان با استاندارد LSB منطبق است.

آیندهٔ LSB

LSB تاکنون تلاش کرده است مشخصات تمامی آنچه را که مربوط به لینوکس می‌شود، پوشش دهد. آینده‌های LSB یکی از کمیته‌های فرعی گروه کاری LSB است. هدف آن گسترش محدوده‌های کاری LSB است. اگر مایلید LSB را گسترش دهید باید با این گروه کار کنید.

بین‌المللی‌سازی لینوکس

دومین گروه کاری اصلی گروه استانداردهای آزاد، برنامه بین‌المللی‌سازی لینوکس[1] (www.li18ux .org)است. این گروه گروهی از APIهای اصلی لینوکس را استاندارد می‌کند به گونه‌ای که برنامه‌های بومی شده به راحتی بر روی لینوکسهای LSB اجرا شوند.

[1] Linux Internationalization Initiative

فروشندگان توزیعهای لینوکس از استانداردهای li18nux[1] بهره می‌برند بدین ترتیب که اطمینان دارند که توزیع آنها در هر نقطه‌ای از دنیا کار می‌کند و می‌تواند برنامه‌های بومی شده هر منطقه را اجرا کند.

شرکتهای بزرگ و تولیدکنندگان نرم‌افزار نیز با استفاده از این استاندارد، می‌توانند نرم‌افزار خود را متناسب با مشخصات بومی هر منطقه‌ای طراحی کنند.

در اینجا به شرح چند اصطلاح می‌پردازیم:

- **بین‌المللی‌سازی** – فرآیند طراحی یا اصلاح یک برنامه یا بستر نرم‌افزاری، به گونه‌ای که قابل بومی شدن باشد.
- **بومی سازی** فرآیند پشتیبانی از مشخصات فرهنگی از قبیل زبان، واحد پول در یک برنامه. پس از اینکه برنامه‌ای بین‌المللی شد، می‌توان آن را برای هر منطقه جغرافیایی بومی سازی کرد.
- **جهانی سازی** – ترکیب بین‌المللی‌سازی و بومی‌سازی

اگر قصد دارید برنامه‌ای برای کاربرد داخل سازمان یا فروش تجاری بنویسید، باید خود را ملزم به بین‌المللی‌سازی آن کنید. با اینکار تضمین می‌کنید که برنامه شما در صورت نیاز برای هر کشوری قابل بومی سازی است. برخی شرکتها وجود دارند که متخصص ترجمه و بومی‌سازی هستند و در صورت نیاز به بومی‌سازی برنامه، می‌توان از تخصص آنها بهره برد.

Li18nux تعدادی API در اختیار شما قرار می‌دهد که می‌توانند با مجموعه کاراکترهای لاتین، آسیایی و تعداد زیادی از کشورهای دیگر کار کنند. همچنین ابزارهایی یرای آزمایش انطباق برنامه با li18nux در اختیار قرار می‌دهد. با li18nux بدون نیاز به تغییر کد برنامه خود می‌توانید قابلیت پشتیبانی از هر زبانی را به آن اضافه کنید.

[1] عدد ۱۸ در li18nux نماینده ۱۸ حرف بین i و n در کلمه internationalization است.

آزمایش و تطابق

گروه استانداردهای آزاد از هر فرصتی برای مستندسازی آنچه که موجود است استفاده می‌کند. جنبه‌های آزمایش و انطباق نیز از این قاعده مستثنی نیستند. در دنیای یونیکس، "گروه باز[1]" برای اطمینان از تطابق برنامه‌ها با استانداردها. آزمایشهای گسترده‌ای برروی آنها انجام می‌دهد. گروه استانداردهای آزاد با گروه باز و سایر گروه‌های حرفه‌ای در زمینه لینوکس تعامل دارد و بجای اینکه از اول شروع کند، از تجربه آنها برای طراحی آزمایشهای خود بهره می‌برد. مجموعه آزمایشهای تطابق با LSB به ۴ دسته تقسیم می‌شوند:

- **آزمایش توزیعهای لینوکس** – این بخش از آزمایش‌ها مخصوص فروشندگان توزیعهای لینوکس است. ممکن است شما بخواهید لینوکس مورد استفاده سازمان خود را در داخل سازمان طراحی کنید. در حالت هم باید توزیع شما مطابق با LSB طراحی شده باشد تا بتوانید برنامه‌های مطابق با LSB را روی آن اجرا کنید.
- **آزمایش برنامه‌ها** این مهمترین آزمایش برای تولیدکنندگان نرم‌افزار و خریداران شرکتی می‌باشد. حتی اگر برنامه‌های شما برای استفاده داخل سازمان نوشته می‌شوند باید مطابق با LSB نوشته شوند. این کار تضمین می‌کند که برنامه شما برروی هر توزیع مطابق با LSB کار می‌کند.
- **آزمایش محیط ساخت** – این آزمایش هم مخصوص تولیدکنندگان نرم‌افزار است. این آزمایش محیط ساختن (نوشتن) برنامه را از نظر داشتن تمامی امکانات مورد نیاز LSB، تحت آزمایش قرار می‌دهد.
- **آزمایش نمونه پیاده‌سازی شده** – این آزمایش، روش نهایی آزمایش تطابق با LSB است. در این روش یک محیط اجرای برنامه ارایه می‌شود که نوعی سیستم [عامل] حداقل و به

[1] Open Group

طور خالص LSB می‌باشد. تولیدکنندگان نرم‌افزار می‌توانند برنامه‌های خود را تحت این سیستم آزمایش کنند.

li18ux تعدادی آزمایش تطابق دیگر نیز دارد که می‌توانید آنها را در سایت اینترنتی آن بیابید. مجموعه تستهای li18ux به دو صورت وجود دارند: تعاملی که اجزای قابل دیدن برنامه را آزمایش می‌کنند و خودکار که پیاده‌سازی درست رابطهای بین‌المللی‌سازی را آزمایش می‌کنند.

در زمان نوشتن این کتاب، گروه استانداردهای آزاد در حال آغاز برنامه‌ای برای دادن تاییدیه به فروشندگان توزیعها و نرم‌افزارها بود که بتوان آن را به صورت برچسبی برروی بسته‌بندی محصول چسباند. برای اطلاع از این برنامه باید به سایت آنها مراجعه کنید.

توزیعهای تخصصی لینوکس

استانداردهایی که توسط گروه استانداردهای آزاد توسعه یافته‌اند بیشتر بر توزیعهای لینوکس قابل استفاده برای عموم متمرکز شده‌اند. نباید فراموش کرد که یکی از قابلیتهای بزرگ لینوکس قابلیت سفارشی کردن آن (تغییر متناسب با نیاز) می‌باشد. به همین دلیل کاربردهای بسیار زیادی از لینوکس وجود دارد (ست تاپ باکس، PDA، استفاده‌های توکار و در این کاربردها چنان لینوکس را تغییر می‌دهند که دیگر نمی‌توان آن را استاندارد دانست. خوشبختانه با فراگیر شدن این کاربردها تلاشهایی برای استاندارد کردن این محیطها آغاز خواهد شد.

جمع بندی

در این بخش با استانداردهای دنیای فراگیر لینوکس آشنا شدید. با استفاده از توزیعها و برنامه‌های مطابق با LSB مطمئن باشید که هیچگاه در دام یک فروشنده توزیع یا نرم‌افزار گرفتار نخواهید شد. در زمینه شرکتهای بزرگ خودکفا از نظر نرم‌افزار و تولیدکنندگان نرم‌افزار نیز پیروی از li18ux آنها را قادر می‌سازد محصولات خود را با کمترین هزینه در بازارهای کشورهای دیگر عرضه کنند.

فصل ۸

عملیات:

به کارگیری لینوکس و متن‌باز

برای تحقق پیاده‌سازی لینوکس در یک محیط، باید آن را یکی از ملزومات انجام کارهای روزمره قرار داد. در اکثر کسب و کارها لازم است برنامه‌ای برای به کارگیری لینوکس در ساختار موجود تدوین کرد. کسانی که قصد دارند از محیط ویندوز یا یونیکس به لینوکس مهاجرت کنند، در ابتدا باید مهاجرت را درست درک کرده باشند و برنامه‌ای برای این مهاجرت داشته باشند. در ادامه این فصل در زمینه پشتیبانی و آموزش لینوکس بحث خواهد شد.

در این فصل موارد عملی مورد بحث قرار خواهند گرفت. از آنجایی که ساختار سازمانها با هم فرق می‌کند، این فصل را باید به عنوان ابزاری کمکی برای تصمیم‌گیری در زمینه به کارگیری لینوکس تلقی کرد. در اینجا یک مورد خاص مهاجرت از یونیکس یا ویندوز یا هر سیستم‌عامل دیگری به لینوکس را بررسی نمی‌کنیم، بلکه ملاحظات مهاجرت به لینوکس معرفی خواهند شد. قابل ذکر است در مواردی مهاجرت به لینوکس ممکن است صلاح نباشد و نباید انجام شود.

پیاده سازی

قبل از پیاده‌سازی لینوکس در یک سازمان، باید طرح جامع این کار از قبل طرح‌ریزی شده باشد و مشخص باشد که کدام برنامه‌ها را باید تحت لینوکس پیاده‌سازی کرد. در اکثر موارد باید لینوکس را در کنار ساختار دیگری نصب کرد. حتی اگر قصد داشته باشید در تمام زیرساخت

سازمان خود از لینوکس استفاده کنید، رسیدن به این هدف ممکن است ماه‌ها یا حتی سالها طول بکشد.

برای پیاده‌سازی لینوکس در یک سازمان در ابتدا باید مشخص باشد که از کدام توزیع آن و با چه پیکربندی قرار است استفاده شود. همان گونه که در فصل مربوط به توزیعها ذکر شد، در اغلب موارد باید یک توزیع سفارشی برای خود تهیه کنید. بسته نرم‌افزاری لینوکس سفارشی باید به گونه‌ای باشد که از عهده پشتیبانی آن برآیید. در قدم بعد باید برنامه‌ها و ابزار تجاری یا متن‌باز موجود برای لینوکس را بررسی کنید و آنچه را که مناسب نیازهای شماست بیابید. پس از یافتن برنامه‌ها بعد آنها را به خوبی آزمایش کنید و از کارایی آن اطمینان حاصل کنید.

اکثر فروشندگان توزیعهای لینوکس خدماتی را نیز ارائه می‌دهند که از طریق آنها می‌توان لینوکس تهیه شده را به روز کرد. برخی از این خدمات هزینه‌بر هستند. این گونه خدمات برای کسب و کارهای کوچک یا کاربران خانگی بسیار مفید هستند. اما در مورد اکثر شرکتهای متوسط و بزرگ، این گونه خدمات را باید متناسب با محیط اطلاعاتی آن شرکت ارائه کرد. بنابراین باید با توزیع‌کننده لینوکس در زمینه نیازهای خود مذاکره کرد.

به کارگیری لینوکس در سازمانهای مختلف متفاوت است و نمی‌توان تمامی انواع سازمانها را در این فصل گنجاند. با این حال مواردی وجود دارد که هر سازمانی قبل از مهاجرت یا اصلاح وضعیت موجود باید در نظر بگیرد.

پیاده‌سازی جغرافیایی

اگر سازمان شما در کشورهای مختلف دفاتری داشته باشد، باید توزیعی از لینوکس را انتخاب کنید که بتوان آن را در مقیاس جهانی به کار گرفت. در فصل ۵ آموختیم که اکثر توزیعها جهانی هستند. این احتمال وجود دارد که کارمندان شما در این دفاتر با توزیعهای محلی لینوکس آشنا باشند و حتی آن را برای استفاده در سازمان توصیه کنند. این احتمال هم وجود دارد که آن توزیع لینوکس تنها توزیعی باشد که برای آن منطقه بومی‌سازی شده است.

شما می‌توانید یک توزیع را برای استفاده در تمامی دفترها انتخاب کنید یا دفاتر مختلف را در انتخاب توزیع مورد نظر خود آزاد گذارید. در اینجا یکی از زیباییهای استاندارد LSB نمایان می‌شود. اگر برنامه‌های کاربردی شما با استاندارد LSB سازگار باشند، در تمامی توزیعهای لینوکس در هر جای دنیا می‌توان از آنها استفاده کرد.

مهاجرت و هم‌زیستی

همان گونه که در ابتدای فصل اشاره شد، ما به موارد خاص اشاره نخواهیم کرد، بلکه با نگاهی علمی به این موضوع خواهیم پرداخت. منابع زیادی در این زمینه در اینترنت موجود است که برخی از آنها در بخش مراجع این کتاب آورده شده است. می‌توان یک کتاب را به جزئیات مسائل مطرح در مهاجرت از بسترهای سخت‌افزاری گرفته تا برنامه‌های کاربردی، اختصاص داد. اما در اینجا مسائل مهم مطرح در مهاجرت در لایه سیستم‌عامل را بررسی خواهیم کرد.

مهاجرت قطعاً هزینه‌بر است. برای اینکه این هزینه‌ها قابل توجیه باشند باید مزایا و صرفه‌جوئی‌های ناشی از مهاجرت را نیز در نظر گرفت. به همین دلیل است که اگر سیستم کنونی شما قابل اطمینان باشد و نیازهای تجاری شما را با حداقل هزینه برآورده کند، مهاجرت توجیه‌پذیر نخواهد بود. در ادامه برخی از هزینه‌های عمده مهاجرت را برمی‌شماریم:

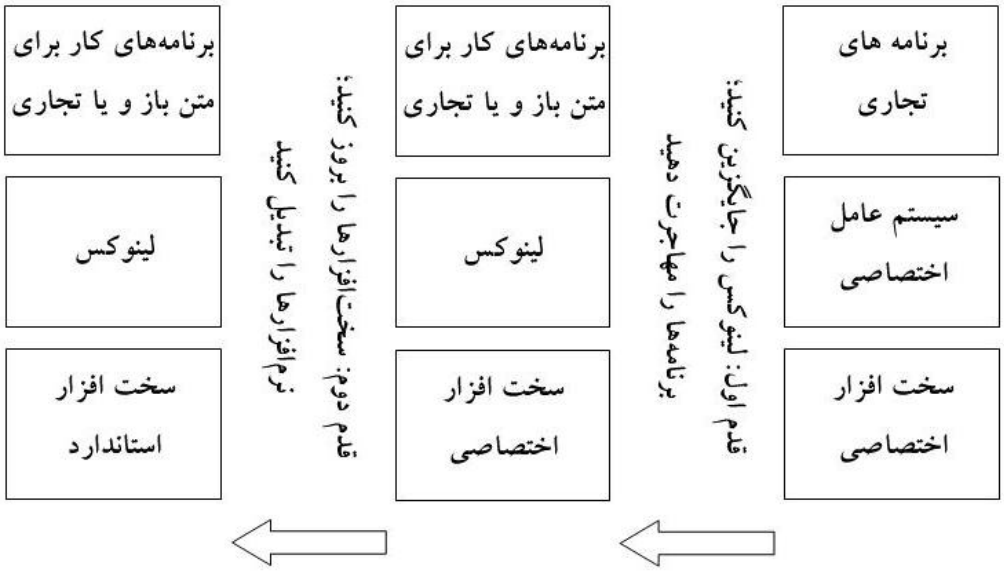
- **سخت‌افزار**- مهاجرت به لینوکس ممکن است نیازمند تغییر بستر سخت‌افزاری باشد. بسته به اینکه چقدر از عمر سخت‌افزار موجود گذشته باشد، این گونه تغییرات ممکن است منطقی باشد. از طرف دیگر همان گونه که در بحث‌های بعدی خواهید دید قابلیت لینوکس در پشتیبانی از بسترهای سخت‌افزاری گوناگون باعث کاهش چشم‌گیر هزینه‌های سخت‌افزار می‌شود.
- **آموزش**- اگر سیستم کنونی شما عمدتاً بر اساس یونیکس بنا شده باشد، نیاز شما به آموزش کارمندان به حداقل خواهد رسید. اما اگر از ویندوز، ناول یا محیط دیگری به لینوکس مهاجرت می‌کنید، این هزینه‌ها قابل توجه خواهد بود.

- **مجوزدهی**- بسته به اینکه از چه نوع برنامه کاربردی در سازمان خود استفاده می‌کنید، ممکن است نیازمند ارتقاء یا خرید مجدد برخی از این گونه برنامه‌ها باشید، البته با این فرض که تمامی نرم‌افزارهای مورد نیاز شما تحت لینوکس موجود باشد. از طرف دیگر از آنجایی که هزینهٔ تهیهٔ اغلب توزیعهای لینوکس رایگان است، می‌توان به بودجه تهیه برنامه‌های کاربردی افزود.
- **اطلاعات**- دارایی واقعی سازمانها اطلاعات آنهاست، نه سخت‌افزار و نرم‌افزاری که برای ذخیره و بازیابی آن اطلاعات به کار گرفته می‌شود. در انتقال از یک محیط به محیط دیگر، بیشترین هزینه و خطر مربوط به مهاجرت دادن اطلاعات سازمان می‌باشد. در این زمینه در بخشهای بعدی بحث خواهد شد.

برای اینکه درک بهتری از تأثیرات مهاجرت داشته باشیم نیازمند بررسی جزئیات بخشهای مختلف هزینه‌بر در این فرآیند هستیم.

سخت‌افزار

برخی افراد معتقدند، از آنجایی که لینوکس می‌تواند در بسترهای مختلف سخت‌افزاری اجرا شود، بنابراین بر روی تمامی این بسترها نیز به خوبی جواب می‌دهد. در فصل ۲ دلایلی را آوردیم مبنی بر اینکه پیاده‌سازی لینوکس در بسترهایی غیر از بستر اصلی زیرساخت اطلاعاتی سازمان، ممکن است آثار سوئی داشته باشد. با این حال استفاده مجدد از یک سخت‌افزار در سیستمهای غیرمرکزی زیرساخت اطلاعاتی، می‌تواند مناسب باشد. فرض کنید بر روی سخت‌افزار کنونی شما نرم‌افزارهای اختصاصی پیاده شده‌اند. هدف نهایی شما این است که لینوکس را بر روی سخت‌افزارهای استاندارد پیاده‌سازی کنید. اما چنین پیاده‌سازی را نمی‌توان یک شبه انجام داد. اما می‌توان قدمهایی در زمینه استفاده مجدد از سخت‌افزارهای موجود برداشت و فرآیند مهاجرت را در طی چند ماه تکمیل کرد.



شکل ۸-۱: مهاجرت کردن به لینوکس و سخت‌افزارهای استاندارد

شکل ۸-۱ یک تصویری ساده از مراحل انتقال از یک محیط کاملاً اختصاصی به محیطی استاندارد و باز در اختیار می‌گذارد. مسلماً حجم کار مورد نیاز برای هر یک از این مراحل می‌تواند بسیار زیاد و پیچیده باشد. در اینجا هم اگر محیط اختصاصی موجود، یونیکس باشد، تبدیل برنامه‌ها، فایلهای دستور و سایر اجزاء زیرساخت نرم‌افزاری به حداقل می‌رسد. سیستم‌عاملهای غیریونیکس دیگر مانند VMS، MVS و ... نیازمند بازنگری عمیق قبل از استفاده در لینوکس هستند. یکی از مشکلات مرحله اول، نرم‌افزارهای تجاری هستند. زیرا اکثر تولیدکنندگان این گونه نرم‌افزارها برنامه‌های خود را فقط تحت بسترهای سخت‌افزاری استاندارد ارائه می‌کنند و ممکن است دریافت یک نسخه از آن نرم‌افزار تحت یک سخت‌افزار خاص غیرممکن باشد. در چنین شرایطی شما باید ابتدا بر انتقال برنامه‌ها و فایلهای دستوری که خودتان نوشته‌اید، تمرکز کنید و برنامه‌های تجاری را در مرحله نهایی انتقال دهید.

در مرحله دوم تغییر سخت افزار، باید شرایط محیطی را بررسی کنید. آیا محیط کار شما برق، ابزار خنک کننده، فضا و سایر وسایل مورد نیاز برای سخت‌افزارهای جدید را دارد؟ مسائل ارتباطی را نیز باید درنظر بگیرید. در مواردی شاید لازم باشد از وسایل ورودی و خروجی موجود برای

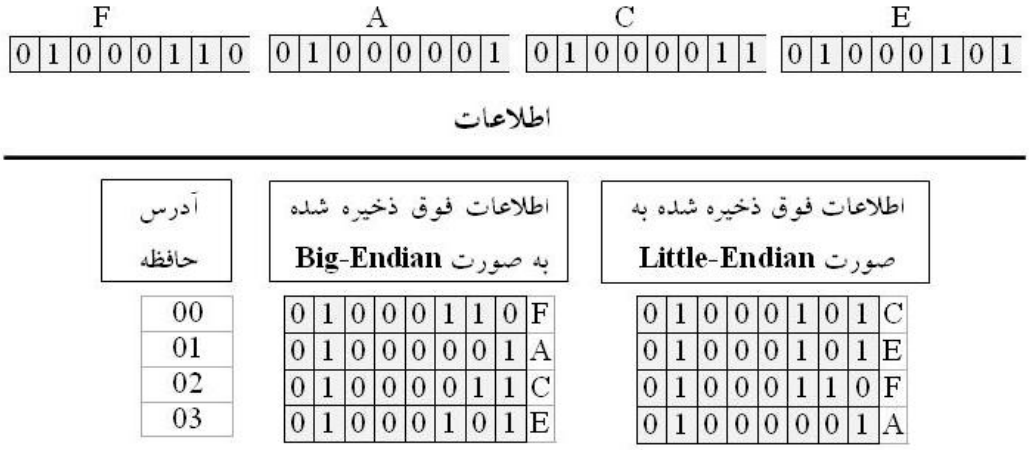
سخت‌افزار جدید استفاده کنید. تمامی اتصالات شبکه موجود را نیز باید بررسی کنید. این نکته را نیز در نظر داشته باشید که وقتی سخت‌افزار را عوض می‌کنید خواص مربوط به کارآیی و قابلیت اطمینان و ارتقاء سخت‌افزار نیز تغییر می‌کند (ممکن است بهتر یا بدتر شود). باید در زمینه تأثیر این عوامل بر کاربر و کسب و کارتان بررسی دقیقی انجام دهید.

اطلاعات

درست است اطلاعات نیاز به بررسی بسیار دقیقی دارد، بخصوص اگر این اطلاعات میان سیستمهای مختلف به اشتراک گذاشته شده باشد. در مهاجرت اطلاعات درگیر مسائلی چون شیوه ذخیره اطلاعات، به اشتراک گذاری آن بین سیستمهای مختلف و اطلاعات برنامه‌های کاربردی خواهید بود. اولین نکته‌ای که باید در نظر گرفته شود، ترتیب بایت Endian می‌باشد.

Endian

جاناتان سوئیفت در کتاب «سفرهای گالیور» توضیح می‌دهد که ساکنانی از لی‌لی‌پوت که تخم‌مرغ را از سمت بزرگ آن می‌شکنند، Big Endian نام دارند. در مقابل به کسانی که تخم‌مرغ را از سمت کوچک آن می‌شکنند، Little Endian گفته می‌شود. عبارت Endian را صنعت کامپیوتر برای تمایز بین ترتیب‌های مختلف بایت در سیستمهای مختلف، به عاریه گرفته است. ترتیب بایت Endian که در شکل ۸-۲ نمایش داده شده است، ترتیب قرار گرفتن بایتها موقع ذخیره و بازیابی اطلاعات، در یک سیستم را تعیین می‌کند. اگر اطلاعات به صورت Big Endian ذخیره شده باشند، و سیستم دیگری این اطلاعات را به صورت Little Endian بازیابی کند، اطلاعات بازیابی شده قابل استفاده نخواهد بود و باید با برنامه‌ریزی درست از این خطا جلوگیری کرد.



شکل ۸-۲: ترتیب قرارگیری بایتهای اطلاعات در سیستمهای **Endian**

هر سیستم‌عامل یا ریزپردازنده‌ای ممکن است ترتیب بایت مخصوص خود را داشته باشد، لینوکس در معماریهای IA-32 و IA-64 اطلاعات را به صورت Little Endian ذخیره می‌کند. اما در معماریهای SPARC و PA-RISC به صورت Big Endian عمل می‌کند. سیستم‌عامل HP-UX تحت PA-RISC و سولاریس تحت SPARC به صورت Big Endian هستند در حالی که سولاریس تحت IA-32 به صورت Little Endian عمل می‌کند. این مشکل ترتیب بایت مهمترین مشکل در فرآیند مهاجرت است که به آن به اندازه کافی توجه نمی‌شود.

همان گونه که می‌بینید، لینوکس در زمینه ترتیب بایتها به صورت دوگانه عمل می‌کند. این دوگانگی به نوع معماری سخت‌افزاری بستگی دارد. به همین دلیل است که برنامه‌نویسان لینوکس برنامه‌ای را که به مسأله Endian توجه کافی ندارد، ناقص تلقی می‌کنند. جامعه برنامه‌نویسان لینوکس، مجموعه‌ای از ابزار را طراحی کرده است که به کمک آنها می‌توان این ضعف را در برنامه‌ها تشخیص داد. این یکی از دلایلی است بر این گفته که برنامه‌های لینوکس به راحتی در معماریهای مختلف کار می‌کند.

اکثر برنامه‌های تجاری که تحت معماریهای مختلف ارائه شده‌اند، با این مشکل دست به گریبان بوده‌اند. این بدین معناست که اطلاعات این گونه برنامه‌ها از نظر Endian مشکلی ندارد و شما می‌توانید اطلاعات ذخیره شده تحت یک سیستم اختصاصی را در یک سیستم لینوکس بازیابی

کنید. اما در زمینه برنامه‌هایی که داخل سازمان طراحی شده‌اند باید بسیار دقت کرد، زیرا موقع طراحی اغلب این برنامه‌ها، مسائل مربوط به Endian، در نظر گرفته نشده است. بنابراین موقع انتقال اطلاعات این برنامه‌ها باید دقت کافی را لحاظ کرد.

فایل سیستمها

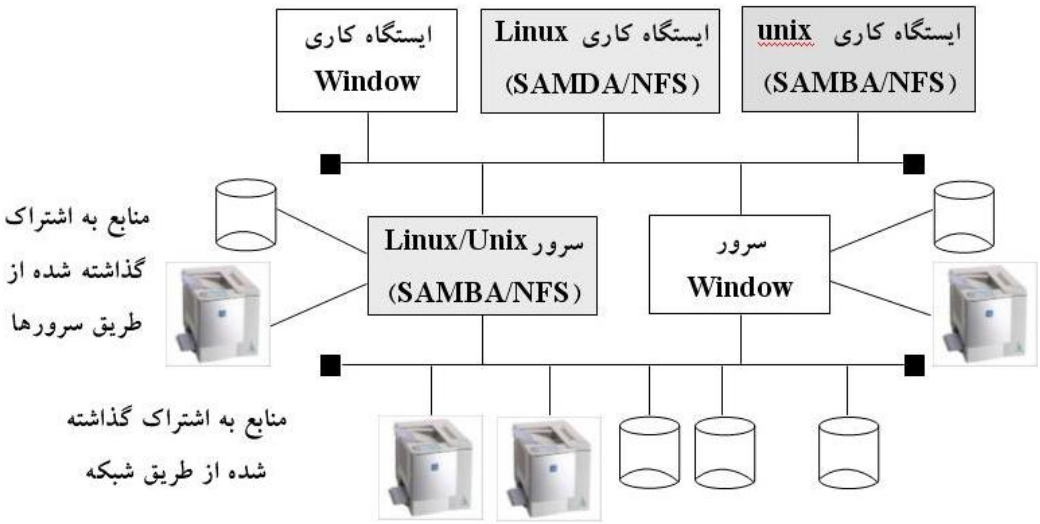
در فصلهای قبلی برخی از فایل سیستمهای موجود در لینوکس را بررسی کردیم. متداولترین آنها در لینوکس ext2 و ext3 می‌باشد. اگر از سیستمهای یونیکس مهاجرت می‌کنید، باید بدانید که در حال حاضر از چه فایل سیستمی استفاده می‌شود و آیا این فایل سیستم در لینوکس نیز قابل استفاده است. اگر از سیستمهای ویندوز مهاجرت می‌کنید یا همزمان از آنها استفاده می‌کنید، بهتر است اطلاعات به یکی از فایل سیستمهای لینوکس منتقل شود. خوشبختانه جامعه متن‌باز درایورهایی برای تمامی فایل سیستمهای میکروسافت طراحی کرده است. بنابراین در زمینه دسترسی به فایلهایی که در سیستمهای ویندوز وجود دارند، نگرانی وجود نخواهد داشت.

در حال حاضر فایل سیستمهای تجاری و باز بسیار زیادی وجود دارد که نمی‌توان آنها را در این بحث گنجاند. جامعه متن‌باز درایور اکثر این فایل سیستمها را طراحی کرده است. بسیاری از شرکت‌های تجاری که فایل سیستمهای خود را طراحی کرده‌اند، نسخه‌هایی از آن را نیز تحت لینوکس ارائه کرده‌اند.

به اشتراک گذاری

مهمترین نکته‌ای که بعد از طرح فایل سیستمها مطرح می‌شود، سیستمهای به اشتراک‌گذاری فایلها و پرینترهاست. این گونه سیستمها به کاربران یک شبکه اجازه می‌دهد فایلها، پرینترها و سایر منابع خود را در شبکه به اشتراک گذارند. به کمک این سیستمها کاربران می‌توانند از منابع گران‌قیمت موجود به طور مؤثری استفاده کنند. متداولترین راه به اشتراک‌گذاری فایلها و پرینترها،

استفاده از پروتکل‌های میکروسافت است. این پروتکل‌ها عبارتند از: CIFS[1] و SMB[2]. سیستمهای یونیکس به طور سنتی فایلها را از طریق «سیستم فایلی شبکه» یا NFS به اشتراک می‌گذارند. خوشبختانه جامعه لینوکس و بعضی از شرکت‌های تجاری راه حلهایی برای ارتباط میان این پروتکلهای بسیار مهم ارائه کردند. شکل ۸-۳ نحوه ارتباط این خدمات را در یک شبکه نشان می‌دهد.



شکل ۸-۳: مهاجرت دادن منابع به اشتراک گذاشته شده

سامبا (SAMBA) که از SMB گرفته شده است، یکی از بزرگترین فناوریهای متن‌بازی است که امکان ارتباط لینوکس (و یونیکس) با محیطهای ویندوز را فراهم می‌کند. با کمک این پروتکل شما می‌توانید روند مهاجرت کارگزارها و کامپیوترهای رومیزی را از ویندوز با هر سرعتی که صلاح می‌دانید، ادامه دهید. سامبا هم به صورت دهنده و هم به صورت گیرنده موجود است. یک کارگزار سامبا از طرف سایر کامپیوترها همانند یک کارگزار ویندوز دیده می‌شود که منابعی را به اشتراک گذاشته است.

[1] . Common Internet File System

[2] . Server Message Block

اگر زیرساخت شما وابستگی زیادی به NFS داشته باشد این پروتکل را نیز می‌توان هم تحت لینوکس و هم ویندوز پیاده‌سازی کرد. انواع مختلف پروتکل‌های NFS توسط شرکت‌های ثالث برای ویندوز ارائه شده‌اند.

ترکیب سامبا و NFS این امکان را در اختیار شما قرار می‌دهد که براحتی بین ویندوز، لینوکس، یونیکس و هر سیستم‌عامل دیگری که این پروتکل‌ها را می‌شناسد، ارتباط برقرار کند. بدین ترتیب شما می‌توانید به ترکیبی ایده‌آل و با صرفه از سیستم‌عاملهای مختلف مهاجرت کنید.

مدلهای برنامه نویسی

منظور از مدلهای برنامه‌نویسی مجموعه‌ای از توابع (API) است که برنامه‌نویسان از آنها برای نوشتن برنامه خود و اجرای آن تحت یک سیستم‌عامل خاص استفاده می‌کند. از آنجایی که یونیکس و لینوکس ریشه مشترکی دارند و مدلهای برنامه‌نویسی آنها بسیار مشابه است، برنامه‌نویسان یونیکس به راحتی می‌توانند خود را با مدل برنامه‌نویسی لینوکس وفق دهند.

مدل برنامه‌نویسی ویندوز بر اساس هسته‌ای از APIها به نام Win32 استوار است. این مدل با مدلهای استفاده شده در لینوکس و یونیکس بسیار متفاوت است. به همین دلیل برنامه‌نویسان برای انتقال به لینوکس باید با یک سری مسائل دست و پنجه نرم کنند. اگر تسلط کافی بر مدل برنامه‌نویسی Win32 داشته باشید، درک مدل برنامه‌نویسی لینوکس برای شما می‌تواند بسیار مشکل باشد. اکثر برنامه‌نویسان غیریونیکس نیز با همین مشکل روبرو هستند.

اگر دانش برنامه‌نویسی سازمان بر اساس زیانهای برنامه‌نویسی سطح بالا مانند کارگزارهای جاوا و برنامه‌های بانک اطلاعاتی بنا شده باشد، در این صورت انتقال به یک سیستم‌عامل جدید بسیار راحت‌تر خواهد بود و حتی ممکن است از دید برنامه‌نویسان کاملاً شفاف باشد.

برنامه‌های کاربردی

وقتی که از یک محیط اختصاصی به یک بستر لینوکسی مهاجرت می‌کنید، باید معادلهای برنامه‌های کاربردی کنونی خود را در لینوکس بیابید. این سخن بدین معنا نیست که تمامی برنامه‌های متن‌باز می‌تواند جایگزین برنامه‌های تجاری شود، اما این نکته را نیز باید در نظر داشت که در بسیاری از موارد ما یک برنامه تجاری پرقدرت را نصب می‌کنیم، ولی از حداقل امکانات آن استفاده می‌کنیم. اگر شما از تمامی امکانات برنامه‌ای که نصب کرده‌اید، استفاده نمی‌کنید، به احتمال زیاد برنامهٔ متن‌بازی برای شما وجود خواهد داشت که به نحو بهتری نیازهای شما را برآورده کند. به عنوان مثال اگر از یک بانک اطلاعاتی تجاری برای ذخیره و بازیابی معمولی اطلاعات کارمندان استفاده می‌کنید، پایگاه‌های دادهٔ متن‌بازی همچون MySQL و PostGRES این نیاز شما را بهتر برآورده می‌کند.

محیطهای رومیزی

مهاجرت یک محیط رومیزی به لینوکس باید با دقت زیادی صورت پذیرد. بسیاری معتقدند که محیط رومیزی لینوکس در حال حاضر برای کاربران حرفه‌ای یا مهندسین وارد به محیطهای رومیزی یونیکس، مناسب است. گرچه این باور با گذر زمان تغییر خواهد کرد، ارتقاء محیطهای رومیزی برای یک کاربر معمولی نیازمند آموزش گسترده، در زمینه امکانات خود سیستم‌عامل و مجموعه‌های اداری می‌باشد. اگر قصد دارید تمامی محیطهای رومیزی سازمان را به لینوکس تبدیل کنید، اول باید مجموعه برنامه‌های مورد نیاز کاربران را تهیه کنید. همچنین از اجرای درست برنامه‌های تحت وب، در مرورگرهای لینوکس اطمینان حاصل کنید.

راه‌حل‌های دیگری نیز امروزه موجود است که بر اساس پروژه باز WINE (شبیه‌ساز ویندوز) طراحی شده‌اند. Win32 هسته APIهای سیستم‌عاملهای ویندوز است. WINE با این هدف طراحی شده است که این APIها را، در سیستم‌عامل لینوکس، در اختیار برنامه‌ها قرار دهد. WINE اجازه می‌دهد که تعداد محدودی از برنامه‌های کاربردی تحت ویندوز را بتوان تحت لینوکس اجرا کرد.

شرکتهای تجاری در حال آزمایش و تأیید اعتبار برنامه‌های خود تحت WINE هستند و برخی نیز ابزارهایی مخصوص نصب تحت WINE را ارائه کرده‌اند.

تهیه مجوز و خرید برنامه‌ها

مجوز نرم‌افزارهای تجاری برای اجرا تحت لینوکس اغلب بلاتغییر می‌ماند. برخی از تولیدکنندگان و فروشندگان نرم‌افزار تلاش می‌کنند ارتباطی بین هزینه بستر سخت‌افزاری و سیستم‌عامل آن با هزینه برنامه کاربردی اجرا شده در آن سیستم‌عامل برقرار کند. اما در اکثر موارد این ارتباط، از منطق درستی برخوردار نیست. در حقیقت نسبت قیمت به کارایی سخت‌افزارها در طول سالهای اخیر بسیار کاهش یافته است. در حالی که هزینه نرم‌افزارهای تجاری نه تنها کاهش نیافته، بلکه در مواردی بیشتر هم شده است. قیمت نرم‌افزارهای تجاری بر اساس رقابت موجود و امکاناتی که در اختیار مشتری قرار می‌دهد تعیین می‌شود و اغلب ربطی به سخت‌افزاری که روی آن اجرا می‌شود، ندارد.

اگر قصد دارید از یک سیستم‌عامل اختصاصی به لینوکس مهاجرت کنید، باید با فروشندگان نرم‌افزارهای تجاری خود تماس بگیرید و از موجود بودن آن نرم‌افزارها تحت لینوکس و سازگاری آنها با LSB، اطمینان حاصل کنید.

سپس در زمینه تبدیل مجوز نرم‌افزارهای کنونی خود به لینوکس و هزینه آن مذاکره کنید. همان گونه که در قسمت قبل اشاره شد، در مواردی که بتوانید جایگزین متن‌بازی برای نرم‌افزار کنونی خود بیابید، هزینه خرید نرم‌افزار جدید کاهش خواهد یافت.

اگر قصد دارید برنامه‌های کاربردی متن‌باز را پیاده‌سازی کنید، باید مطمئن شوید که کارمندان از امکان کپی آزادانه این گونه نرم‌افزارها، داخل یا خارج از سازمان، آگاهند. مأمورین خرید شما باید فرق بین نرم‌افزار متن‌باز و اختصاصی را بدانند. در نهایت پس از اینکه نحوهٔ خرید نرم‌افزارهای تجاری و اختصاصی مشخص شد، مسلماً مایل خواهید بود روند تهیه نرم‌افزارهای متن‌باز را نیز تعیین کنید. فروشندگان نرم‌افزارهای متن‌باز باید شما را از قیمت تک تک اقلام

خریداری شده همراه نرم‌افزار، مانند رسانه (لوح فشرده حاوی برنامه)، دفترچه‌های راهنما، پشتیبانی و غیره مطلع نماید. در مواردی ممکن است فروشندگان سخت‌افزار مجموعه‌های لینوکس یا نرم‌افزارهای متن‌باز را در سخت‌افزار خود قرار دهد و بابت این کار از شما هزینه دریافت کند. شما باید از ریز این هزینه‌ها مطلع باشید و در زمینه کارا بودن نرم‌افزار در سازمان تصمیم‌گیری کنید. باید بر روی آموزش مأمورین خرید خود در زمینه نحوه کارِ مدل متن‌باز وقت کافی بگذارید. این در نهایت به نفع شما خواهد بود.

پشتیبانی

نحوه پشتیبانی در لینوکس و متن‌باز ممکن است با سیستم فعلی شما تفاوت بسیاری داشته باشد. خوشبختانه در این زمینه حق انتخاب تماماً با شماست. روشهای مختلف پشتیبانی را می‌توان به سه دسته تقسیم‌بندی کرد:

- هر نرم‌افزار یک شرکت
- تمامی نرم‌افزارها، یک شرکت
- استفاده از پشتیبانی جامعه لینوکس و متن‌باز

از آنجایی که این روشهای پشتیبانی احتمالاً با روشهای موجود شما بسیار فرق می‌کند، هر یک از این روشها را با جزئیات بیشتری بررسی می‌کنیم.

هر نرم‌افزار یک شرکت

در این روش برای هر جزئی از مجموعه نرم‌افزارهای استفاده شده با یک شرکت پشتیبانی‌کننده قرارداد بسته می‌شود. معمولاً پشتیبانی سخت‌افزار را فروشنده آن و پشتیبانی سیستم‌عامل را توزیع‌کننده آن ارائه می‌دهد. پشتیبانی هر یک از نرم‌افزارها را نیز ممکن است فروشنده آنها ارائه دهد.

بزرگترین مزیت این روش این است که اگر سازمان از عملکرد هر یک از شرکتها ناراضی باشد، می‌تواند آن نرم‌افزار یا شرکت پشتیبانی کننده را عوض کند، بدون اینکه تأثیری بر پشتیبانی سایر نرم‌افزارها بگذارد. مزیت دیگر این روش این است که می‌توان برای هر یک از برنامه‌ها بهترین پشتیبانی را تأمین کرد.

این روش پشتیبانی برای محصولات متن‌باز تجاری و اسم و رسم‌دار مناسب‌تر به نظر می‌رسد، مانند توزیعهای لینوکس. به عنوان مثال اگر شما توزیع لینوکس خود را از یکی از فروشندگان معتبر دریافت کنید، آن فروشنده در بهترین موقعیت برای رفع مشکلات شما و اصلاح توزیع متناسب با نیازهای شما می‌باشد. حتی در برخی از موارد شما می‌توانید از رقابت ایجاد شده در ارائه خدمات یک محصول سود ببرید. آنچه که گفته شد، می‌توان به تمامی محصولات متن‌باز تعمیم داد.

با وجود اینکه بسیاری از شرکتهای استفاده کننده از لینوکس و متن‌باز این مدل را ترجیح داده‌اند، اما معایبی نیز دارد:

- **تداخل بین مجموعه برنامه‌های مورد استفاده**- همان گونه که گفته شد استفاده از این روش می‌تواند بهترین خدمات را به ارمغان آورد. با این حال در مواردی شما به مشکلاتی برمی‌خورید که منبع آنها را نمی‌دانید. در نهایت مجبور می‌شوید تداخلهای موجود بین پشتیبانی‌کننده‌ها را بررسی و برطرف کنید.
- **ارتقاء نرم‌افزارها**- مدیریت و نظارت بر عملکرد فروشنده‌ها و هر یک از قراردادهای پشتیبانی بدین معناست که شما مالکیت نرم‌افزارهایتان را در اختیار خود دارید. اگر یکی از این نرم‌افزارها را ارتقاء دهید، ممکن است شرایطی بوجود آید باعث لغو قرارداد پشتیبانی دیگر شود. بنابراین در این روش باید شرایط پشتیبانی هر یک از فروشندگان در قرارداد آنها به خوبی مشخص باشد و موارد احتمالی تداخل ایجاد شده بین نرم‌افزارها و نحوه برطرف کردن آن در نظر گرفته شده باشد.

- **مدیریت قراردادها-** در این روش نظارت دقیق بر هر یک از قراردادهای پشتیبانی الزامیست. سهل‌انگاری در این زمینه به شکست این روش پشتیبانی در سازمان، منجر خواهد شد.

بنابراین معایب این روش را همانند مزایایش به خوبی در نظر بگیرید و با دید بهتری این روش را انتخاب یا رد کنید.

تمامی نرم‌افزارها، یک شرکت

این روش نقطه مقابل «هر نرم‌افزار، یک شرکت» قطب مخالف هر نرم‌افزار یک شرکت می‌باشد. در این روش، از پشتیبانی یک شرکت برای تمامی نرم‌افزارهای خود بهره می‌برید. مزیت این روش واضح است: شما با یک شرکت طرف حساب خواهید بود. در این روش شرکت مورد نظر، مجموعی از خدمات را باید در اختیار شما قرار دهد. از آنجایی که محصولات پشتیبانی شده متن‌باز هستند، این شرکت می‌تواند نقش اصلاح کد آنها را نیز در موارد برخورد با مشکل، بر عهده گیرد. در روش «تمامی نرم‌افزارها، یک شرکت» این امکان وجود دارد که شرکت طرف حساب شما، پیمانکار اولیه باشد و پشتیبانی هر یک از اجزاء را به پیمانکارهای فرعی واگذار کند. صرف نظر از تصمیم پیمانکار وی باید از نقاط تأثیرگذار در جامعه متن‌باز باشد و صلاحیت اصلاح کد برنامه‌ها را احراز کرده باشد.

استفاده از پشتیبانی جامعه لینوکس و متن‌باز

سازمانهای فناوری اطلاعات بزرگ، ممکن است به این نتیجه برسند که خودشان محصولات استفاده شده را پشتیبانی کنند و در موارد خاص از شرکتهای تجاری بهره گیرند و بدین ترتیب از هزینه‌های پشتیبانی بکاهند. در این روش یک تیم فنی خِبره و دانش‌آموخته مورد نیاز است. برای ایجاد چنین تیمی، باید آنها را در برقراری ارتباط با بازیگران اصلی جامعه متن‌باز آزاد گذاشت. هدف شما باید این باشد که نیازهای پشتیبانی خود را برآورده کنید و در عین حال می‌توانید به پیشرفت جامعه متن‌باز نیز کمک کنید. در این زمینه نباید از ابتدا انتظار داشته باشید همانقدر که به جامعه

متن‌باز خدمت می‌کنید، از آن بهره‌برداری کنید، بلکه با کمی صبر ثمره آن را خواهید دید. در این روش، بین دو طرف (سازمان و جامعه متن‌باز) وابستگی ایجاد می‌شود و از آن سود می‌برند.

اگر پشتیبانی داخلی با پشتیبانی تجاری همراه شود، در این صورت نتیجه این خواهد شد که کارمندان شما نقش اصلی را در یافتن مشکلات و برطرف کردن آنها ایفا خواهند کرد. پس از یافتن مشکل، اغلب باید کد منبع اصلاح شود و اصلاحات برای نگهدارنده اصلی کد منبع فرستاده ارسالشود. سپس در صورتی که از خدمات تجاری نیز استفاده می‌کنید، آنها را از اصلاحات انجام شده در برنامه مطلع سازید تا در نسخه‌های بعدی از آنها بهره گیرند. همچنین در قرارداد خدمات پشتیبانی تجاری باید شرکت پشتیبانی کننده ملزم به اعمال کردن اصلاحات برنامه تا زمان عرضه نسخه بعدی باشند.

مزیت اصلی این روش پشتیبانی این است که سازمان شما نقش اصلی را ایفا خواهد کرد و نظارت مستقیمی بر مشکلات ایجاد شده و علت آنها خواهد داشت. در روشهای پشتیبانی تماماً تجاری، شما باید امیدوار باشید که آن شرکت، حیاتی بودن آن مشکل را درک کند. در غیر این صورت باید ضرر ناشی از تأخیر ایجاد شده در رفع مشکل را تقبل کنید. اما در مدل ذکر شده در بالا، تصمیم‌گیر اصلی خودتان هستید، خودتان تصمیم می‌گیرد که آیا برای تغییرات یا اصلاحات سرمایه‌گذاری شود یا نه و در این صورت خودتان مستقیماً با تولیدکننده آن نرم‌افزار در راستای اعمال آن تغییرات کار خواهید کرد. مادامی که تغییرات یا اصلاحات درخواستی شما در راستای حرکت آن پروژه باشد (پروژه‌ای که ماحصل آن، نرم‌افزار مورد استفاده شما بوده است)، مدیران پروژه از اعمال تغییرات شما خوشحال نیز خواهند شد، البته با این فرض که سابقه خوبی در خدمت به جامعه متن‌باز داشته باشید.

تأثیرگذاری و ارتباطات

صرفنظر از روش پشتیبانی که شما انتخاب می‌کنید، رمز موفقیت شما در «تأثیرگذاری و ارتباطات» است. سؤال اولی که در این راستا باید پاسخ داده شود این است که چه کسی قرار است

رابطه مابین سازمان شما و نگهدارندگان کلیدی [کدهای منبع] برقرار کند. شما می‌توانید این وظیفه را به فروشندگان نرم‌افزارها محول کنید یا خودتان آن را برعهده گیرید. راه سوم هم این است که از ترکیبی از این دو روش استفاده کنید. این یکی از مزایای اصلی نرم‌افزارهای متن‌باز است که شما می‌توانید کنترل و نظارت محیط را خودتان برعهده گیرید و در انتها از ساختار خدمات‌رسانی رقابتی موجود نیز بهره گیرید. ممکن است برخی از شرکتها چنان به خدمات تجاری وابسته شده باشند که نتوانند روش خود را عوض کنند تا چه رسد بخواهند از روشی استفاده کنند که نیازمند رابطه مستقیم با جامعه متن‌باز و تأثیرگذاری بر آن باشد. اگر قصد دارید از مزایای متن‌باز بهره گیرید ولی با روش پشتیبانی از طریق تأثیرگذاری مشکل دارید، در این صورت بهترین راه این است که با شرکتهای تجاری کار کنید که سابقه خوبی در ارتباط با جامعه متن‌باز دارند.

آموزش

اکثر فروشندگان سیستم‌عامل برنامه‌هایی برای مدرک دادن به کاربران دارند. به کمک این مدرک‌ها، شرکتها می‌توانند از سطح آشنایی کارمندانشان با برنامه موردنظر آگاهی یابند. امروزه گذراندن آموزشهای استاندارد، یک برگ برنده تلقی می‌گردد. این آموزشها برای فروشنده سیستم‌عامل نیز مزایایی دارد از جمله اینکه متخصصان فناوری اطلاعات در انجام هرچه بهتر نصب و پیکربندی آن سیستم‌عامل با یکدیگر رقابت می‌کنند و وجهه بهتری از آن شرکت به نمایش می‌گذارند. مزیت آن برای فروشگاههای محصولات فناوری اطلاعات نیز این است که معیار بهتری برای استخدام کارمندان و ارزیابی دانش موجود در زمینه پیاده‌سازی و پشتیبانی نرم‌افزارها، در اختیارشان قرار می‌دهد. برای سازمانها نیز این آموزشها، رزومه تحصیلی کارمندان آنها را سنگین‌تر می‌کند که در نهایت به سطح دانش آن سازمان می‌افزاید.

نیاز به استانداردسازی آموزشهای لینوکس مستقل از توزیع کننده، منجر به تأسیس مؤسسه تخصصی لینوکس یا LPI[1] (www.lpi.org) شد. LPI سه مدرک در سه سطح ارائه می‌کند. سطح اول برای کاربران حرفه‌ای و پشتیبانی معمولی لینوکس مناسب است. سطح دوم برای طرح‌ریزی و پیاده‌سازی شبکه‌های کوچک طراحی شده است و سطح سوم نیز برای معماران ارشد فناوری اطلاعات که وظیفه طراحی و پیاده‌سازی زیرساخت شرکتهای بزرگ را بر عهده دارند، در نظر گرفته شده‌اند.

برنامه‌های آموزشی دیگری نیز برای لینوکس وجود دارد، از جمله Comptia (www.comptia.org) و SAIR (www.linuxsertification.org). همچنین برخی از توزیع‌کنندگان لینوکس نیز مدرکهایی برای توزیع خود ارائه می‌کنند. سازمان شما باید جزئیات هر یک از این مدرک‌ها را مطالعه کند و در زمینه پذیرش یا عدم پذیرش هر یک از آنها تصمیم گیرد. این کار، در تهیه برنامه آموزشی کارمندان موجود و در حال استخدام، به شما کمک می‌کند.

در اغلب این برنامه‌های آموزشی، کار با برخی از اجزاء موجود در اکثر لینوکسها، مانند سامبا و آپاچی، آموزش داده می‌شود. اما در هیچ یک از آنها اشاره‌ای به نرم‌افزارهای کاربردی مورد استفاده شما نمی‌شود. بنابراین باید پس از اینکه این نرم‌افزارها را تعیین کردید به دنبال دوره‌های آموزشی آنها برای کارمندان خود باشید. از طرف دیگر شما می‌توانید برنامه آموزش کارمندان را نیز خودتان طراحی کنید. این هم یکی دیگر از مزایای تکنولوژی متن‌باز است. از آنجایی که کد منبع همه پروژه‌ها در دسترس است، شما می‌توانید برنامه‌های آموزشی را تا هر سطحی که صلاح می‌دانید طرح‌ریزی کنید. حتی می‌توانید از این راه کسب درآمد کنید.

در نهایت آموزشی که باقی می‌ماند، یادگیری فلسفه متن‌باز است. این آموزشها شامل درک مجوزها و ملاحظاتی است که کارمندان موقع پیاده‌سازی و استفاده از نرم‌افزارهای متن‌باز باید مدنظر داشته باشند. بخش سوم این کتاب نحوه استفاده از نرم‌افزارهای متن‌باز در یک سازمان را

[1] . Linux Professional Institute

بررسی می‌کند. همچنین نحوه عرضه کردن نرم‌افزارهای داخلی سازمان، خود تحت مجوزهای متن‌باز بررسی خواهد شد. فراموش نکنید که آموزش جزئیات این فرآیندها، مهمترین بخشِ برنامه آموزشی شما می‌باشد.

جمع‌بندی

در این فصل، زیربنای دانش شما در زمینه لینوکس و متن‌باز تکمیل شد. با مطالعه این فصل شما باید به درک عمیقتری از مراحل عملی پیاده‌سازی لینوکس رسیده باشید. مسلماً آسانترین مهاجرت به لینوکس از محیطهای یونیکس می‌باشد. اما مهاجرت از سایر محیطها نیز تاکنون موفقیت‌آمیز بوده است.

اکنون که درک بهتری از لینوکس پیدا کرده‌اید، زمان آن رسیده است که اعلام کنیم لینوکس خود یکی از پروژه‌های بزرگ متن‌باز می‌باشد. همان گونه که تاکنون چندین بار اشاره شد، آنچه تاکنون آموخته‌اید، نه تنها به لینوکس بلکه به تمامی پروژه‌های متن‌باز قابل تعمیم است. در بخش سوم متن‌باز را با جزئیات بیشتری بررسی خواهیم کرد و در زمینه گسترش آموخته‌های لینوکس به هر پروژه متن‌باز دیگری بحث خواهد شد.

بخش سوم

نقش متن‌باز در کسب و کار

در بخش سوم نگاه دقیق‌تری به جنبه های مختلف تجاری متن‌باز می‌اندازیم. کلمه «متن‌باز» طیف گسترده‌ای از مفاهیم را شامل می‌شود از جمله مجوزدهی نرم‌افزارها، فرآیند و فرهنگ تولید آنها و غیره. جا انداختن مفاهیم متن‌باز در یک سازمان امری پیچیده و مشکل است که در صورت تحقق آن، سود قابل توجهی برای آن سازمان به ارمغان خواهد آورد.

در فصل اول این بخش، فرآیند تولید نرم‌افزار متن‌باز در یک سازمان بررسی خواهد شد. این فصل درک بهتری از فرهنگ و روشهای تولید برنامه متن‌باز در اختیار شما قرار می‌دهد. فصل دوم این بخش در زمینه تأثیر اقتصادی نرم‌افزارهای متن‌باز بر ساختارهای سنتی کسب و کار بحث می‌کند. در فصل بعدی نگاه عمیق‌تری به مدلهای اقتصادی متن‌باز خواهیم انداخت. در فصل دوازدهم، استفاده از مجوز متن‌باز برای نرم‌افزارهای موجود و به کارگیری نرم‌افزارهای متن‌باز در تجارت بررسی خواهد شد. آخرین نکته در زمینه متن‌باز، نحوهٔ سنجش منابع انسانی و تأثیر آن بر به خدمت گرفتن بهترین استعدادها می‌باشد.

فصل ۹

بازار شرکتی

من همیشه گفته‌ام: «مسأله اصلی لینوکس نیست، بلکه فلسفه متن‌باز است» یا به عبارت دقیق‌تر فرآیند طراحی نرم‌افزارهای متن‌باز. اریک ریموند، به روش طراحی و تولید نرم‌افزارهای متن‌باز، «بازار» اطلاق کرده است که رمز پیشرفت لینوکس می‌باشد. در این فصل ما روش طراحی بازار را به شرکت‌ها تعمیم می‌دهیم. در این فصل شما مفاهیم زیر را خواهید آموخت:

- تفاوت بین روشهای طراحی «کاتدرال» و «بازار»
- ساختار بازار در یک شرکت از نظر تئوری
- چگونگی بهره‌گیری از روش طراحی بازار در شرکتا

یکی از نکات کلیدی که هنگام خواندن این فصل باید مدنظر داشته باشید این است که لزومی ندارد اطلاعات پروژه‌های خود را در اختیار عموم قرار دهیم. در این فصل فقط روشهای توسعه متن‌باز مورد بحث قرار می‌گیرد. توجه داشته باشید که استفاده از روشهای توسعه متن‌باز به این معنا نیست که باید برنامه‌های خود را تحت مجوزهای آن عرضه کنید. در فصل دوازده به طور کامل نحوهٔ عرضه کد پروژه‌های انجام شده در داخل سازمان را مورد بحث قرار خواهیم داد. بنابراین اگر شما پروژه‌ای را در داخل سازمان با استفاده از روشهای توسعه متن‌باز پیش می‌برید، مجبور نیستید که اطلاعات آن پروژه را در اختیار عموم و تحت مجوزهای متن‌باز منتشر کنید.

تشریح راهکارهای توسعه متن‌باز برای یک مدیر دوره‌ای ممکن است بسیار مشکل باشد، بنابراین در این فصل ابتدا ساختار فرضی یک سازمان را تعیین می‌کنیم. این کار دو مزیت دارد: ۱) روشهای طراحی به عباراتی ساده‌تر و آشناتر معرفی می‌شوند. ۲) شما را مجبور به تأمل در زمینه اجزاء و معماری نرم‌افزار موجود در سازمان می‌کند.

کاتدرال و بازار

این مفاهیم از مقاله معروف اریک ریموند گرفته شده‌اند. در این مقاله او روش طراحی سنتی نرم‌افزار را که به خوبی از آن آگاه هستیم، به طراحی و ساخت یک کاتدرال (کلیسای بزرگ) تشبیه می‌کند. در مقابل او روشهای نوین طراحی متن‌باز را که لینوس توروالدز از پیشروان آن است، به روش گسترش بازار تشبیه می‌کند. من قصد ندارم که تمامی مطالب مقاله ریموند را بررسی کنم اما لازم است که درک درستی از روشهای توسعه کاتدرال و بازار داشته باشیم.

- **روش توسعه کاتدرال**- در این روش سنتی، سعی می‌شود که تیمهای کاری کوچک نگه داشته شوند. نحوه طراحی و عملکرد قبل از آغاز ساخت، باید به خوبی مشخص شده باشند. نقشه کامل عملیات از معماری، طراحی تا اجرا و آزمایش به طور دقیق تعیین و مستند شده است. مادامی که طراحی نرم‌افزار کامل نشده است، به بازار عرضه نمی‌شود و نظرات کاربران اکثراً از طریق نسخه‌های آزمایشی عرضه شده آلفا و بتا دریافت می‌شود.
- **روش توسعه بازار**- نگهدارنده یک نرم‌افزار وقتی آن را عرضه می‌کند که حداقل کارایی را داشته باشد و دیگران بتوانند آن را تکمیل کنند، این بدان معناست که نرم‌افزار عرضه شده ناقص بوده و حتی ممکن است ایراداتی نیز داشته باشد. چرخه‌های عرضه نرم‌افزار بسیار زیاد و با فاصله‌های زمانی اندک (حتی یک ساعت) صورت می‌گیرند. بازخوردها بلافاصله بررسی می‌شوند. هر کسی در هر جای دنیا که ایده خوبی برای بهبود برنامه دارد می‌تواند خودش این ایده را به برنامه اضافه کند یا در رفع اشکالات آن تلاش کند. زمان عرضه جدی محصول را نگهدارنده نرم‌افزار به صلاحدید خود تعیین می‌کند.

آنچه که اشاره شد توضیح بسیار مختصری از این دو مفهوم بود. بنابراین توصیه می‌کنم مقاله (یا کتاب ریموند) را خودتان مطالعه کنید. ریموند قصد داشت موفقیت لینوس توروالدز را در زمینه لینوکس تقلید و تکرار کند. او توانست روش توروالدز را که کار کردن با جامعه‌ای از داوطلبان بود، در پروژه دیگری اعمال کند. آنچه که ریموند به آن اشاره نکرده است این است که چگونه می‌توان از

مزایای روش توسعه بازار بهره برد و آنها را در دنیای تجارت به کار گرفت. سؤال اصلی اینجاست که چگونه می‌توان ساختار واقعی شرکتهای بزرگ امروزی را، در برنامه‌ریزیها در نظر گرفت. از جمله بخشهای ساختاری می‌توان به این موارد اشاره کرد: سلسله مراتب مدیریتی، اهداف شغلی کارمندان، قوانین منابع انسانی، در دسترس بودن تجهیزات، سیاستهای راهبردی که توسط هیئت مدیره تعیین شده باشد و فشارهای وارده از طرف شرکتهای رقیب.

ساختار، سیاستها را دنبال می‌کند

قبل از اینکه به سراغ یک مدل عملی برویم، باید اهداف و سیاستها را مشخص کنیم. اشتباه بسیاری از مدیران این است که قبل از تعیین سیاستهای شرکت، ساختار سازمانی را وضع می‌کنند. مسلماً نمی‌توان در این مجال، اهداف و سیاستهای همه شرکتها را در نظر گرفت، اما اهداف مشترکی برای اکثر شرکتها وجود دارد که قابل بحث است. اگر قصد دارید روش توسعه بازار را در شرکت خود پیاده کنید، باید بر تجربه‌های جامعه متن‌باز در استفاده از این روش، مروری داشته باشید:

- **توسعه سریع**- اکثر صاحب نظران بر این باور هستند که جامعه متن‌باز از چرخه‌های تولید قابل توجهی برخوردار است. لینوکس در مدت ۱۰ سال تکامل یافت و تبدیل به یک سیستم‌عامل پیشرفته و قابل اطمینان شد. دو رابط کاربری گرافیکی به نام‌های GNOME و KDE در مدت کمتر از ۲ سال برای آن طراحی شد. البته باید به موفقیت وصف‌ناپذیر پراستفاده‌ترین برنامه متن‌باز، کارگزار وب آپاچی، نیز اشاره کرد.
- **توسعه توزیع شده**- تنها دلیل توسعه صحیح لینوکس و آپاچی را باید توسعه سریع و همزمان اینترنت دانست. جامعه متن‌باز چنان به خوبی از این زیرساخت ارتباطی بهره گرفت که نرم‌افزارها تولید می‌شدند بدون اینکه نویسندگان آنها یکدیگر را دیده باشند. این توانایی متن‌باز در به کارگیری توسعه توزیع شده، آن را قادر می‌ساخت که به بهترین استعدادها صرفنظر از اینکه کجای جهان باشند، دسترسی یابد.

• **بهترین استعدادها-** از مشخصات جامعه متن‌باز این است که کارهای ضعیف را قبول (و حتی تحمل) نمی‌کند. اگر اهل انجام دادن کار اصولی (کد منبع، طراحی، معماری، آزمایش، مستندسازی و ...) نیستید، نمی‌توانید به جامعه متن‌باز راه یابید. البته من تا به حال به عضوی از این جامعه برنخورده‌ام که چنین دیدی داشته باشد، اما یک نوع مدیریت استعداد به طور ذاتی در این جامعه وجود دارد. مدیریت استعدادها از مشکل‌ترین جنبه‌های مدیریت در یک سازمان است.

• **برآوردن نیاز کاربران**- یکی از مهمترین خاصیتهای جامعه متن‌باز این است که برنامه‌نویس، کاربر برنامه نیز هست. بنابراین دور از انتظار نیست که برنامه‌ها متناسب با نیاز کاربران نوشته شود. این مهم باید در سازمان شما پذیرفته شود که برنامه‌نویسان، کاربران برنامه نیز هستند و نظرات آنها، سهم بسزایی در بهبود پروژه خواهد داشت.

• **کیفیت-** ضرب المثل «اگر انسان به اندازه کافی چشم داشت، همه حشرات (باگ، اشکالات برنامه‌ها) ناتوان می‌شدند» را ریموند در مقاله ابداع کرد و امروزه، تبدیل به ضرب المثلی رایج در جامعه متن‌باز شده است. روش توسعه به ظاهر بدون کنترل متن‌باز، منجر به تولید با کیفیت‌ترین نرم‌افزارها شده است. در اینجا منظور از کیفیت این است که یک سیستم بتواند منطبق با تمامی نیازهایی که بخاطر آنها طراحی شده است، عمل کند. مدیران کارکشته پروژه‌های نرم‌افزاری ادعا می‌کنند که برای داشتن کیفیت، باید از ابتدا ساختار پروژه معین و قابل درک باشد. اگر نیاز کاربران فراتر از معماری طراحی شده برای آن سیستم باشد، در این صورت باید اکثر اجزاء سیستم مجدداً طراحی و برنامه‌نویسی شود. اگر بخواهید کیفیت را از معماری به خود برنامه گسترش دهید، یک دید کلی از سیستم باید موجود باشد تا آن را به درستی هدایت کند.

آنچه که در بالا آمد لیست قابل توجهی از مزایای مدل بازار است. برای قانع شدن مدیران، شاید نیازی به ذکر همه این مزیتها نباشد. اما مخالفین این روش توسعه، به برخی از کاستیهای آن اشاره می‌کنند که می‌تواند آن را از موفقیت بازدارد.

- **نیازهای بازار**- این خوب است که برنامه‌نویسان، کاربر برنامه نیز باشند اما اغلب استفاده‌کنندگان از برنامه‌ها، برنامه‌نویس نیستند. در نتیجه، محصول نهایی مطابق با نیازهای واقعی مشتریان نخواهد بود.
- **مسیر توسعه**- اغلب ما، پروژه‌های توسعه را، مانند راه‌هایی می‌دانیم که ارتباطات داخل و خارج سازمان را تسهیل می‌کند. درک این ایده که برنامه‌نویسان وقتی برنامه‌شان را عرضه می‌کنند که آماده شده باشد، برای مدیریت و بخش بازاریابی شرکت مشکل است.
- **مدیریت مردمی**- چه کسی مدیریت این برنامه‌نویسان حرفه‌ای را بر عهده دارد؟ آنها طی چه فرآیندی هدایت می‌شوند؟ مدیریت تیمی از برنامه‌نویسان که استخدام سازمان شما نیستند، کار بسیار مشکلی است.

این کاستیها اجازه نمی‌دهند که مدیر کنترل دلخواه خود را بر پروژه داشته باشد و بتواند در مقال آنچه که عرضه شده است، پاسخگو باشد. آنچه که شما باید یاد بگیرید این است که روشهای کنترل سنتی را کنار بگذارید تا بتوانید محصولات بیشتری با کارایی بالاتر و کیفیتی فراتر از انتظار عرضه کنید. از مزایای طبیعی این گونه عملکرد این است که شما نیروی کاری را در اختیار خواهید گرفت که برای کسب و کار شما بهینه شده‌اند و در عین حال به کار خود عشق می‌ورزند.

بازار ساخت یافته

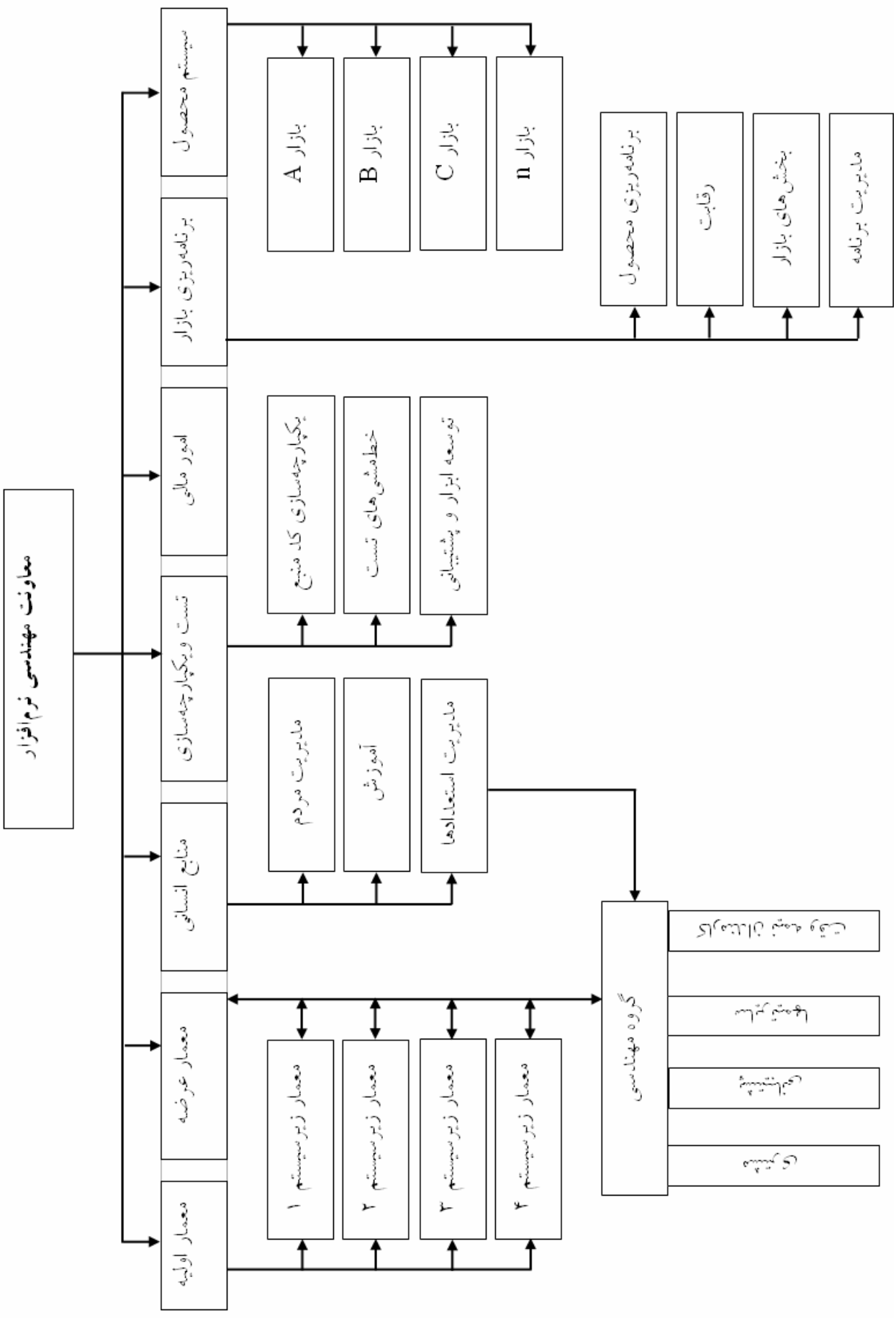
حال که به درک خوبی از اهداف مورد نظر خود رسیده‌اید و از مشکلات احتمالی نیز آگاهی دارید، من می‌توانم یک مدل عملی برایتان ارائه کنم که در شکل ۹-۱ به تصویر کشیده شده است.

مدل عرضه شده یک تیم برنامه‌نویسی بزرگ (بیش از ۱۰۰ مهندس کامپیوتر) را دربرمی‌گیرد. فرقی نمی‌کند که شما در حال تولید یک محصول کوچک تجاری هستند و یا پروژه‌ای بزرگ برای

استفاده در داخل شرکت اجرا می‌کنید. *فراموش نکنید که در این فصل هدف شما این نیست که کد برنامه‌تان را در اختیار عموم قرار دهید، شما فقط می‌خواهید از مزایای توسعه توزیع شده، رایج در جامعه متن‌باز، بهره برید.*

مثال ذکر شده، تصویر جامعی از مراحل توسعه نرم‌افزار به صورت توزیع شده در اختیار شما قرار می‌دهد. شما باید این ساختار عملیاتی را در ساختار سازمانی خود منعکس و اعمال کنید. قبل از اینکه هر بخش از این ساختار را بررسی کنید، سعی کنید بخشهایی از ساختار سازمان خود را که با این ساختار متفاوت است، پیدا کنید. به عنوان مثال یکی از نکات اختلاف با سازمان شما ممکن است این باشد که در این ساختار تمامی دانش مهندسان در یک قسمت متمرکز شده است. همچنین دقت کنید که اکثر فرآیند توسعه پرسنل به جای اینکه در زنجیره مدیریت صورت گیرد، در منابع انسانی انجام می‌شود. این موارد به زودی توجیه خواهند شد. در حالیکه هر بخش از این ساختار را بررسی می‌کنید، در نظر داشته باشید که وظیفه شما این است که یک سیستم‌عامل لینوکس قابل استفاده را پیاده‌سازی کنید (مثلاً به صورت پرتال، بانک اطلاعاتی و یا محصولی خاص). اکنون شما باید بتوانید این دو نکته را به یکدیگر مرتبط کنید که جامعه متن‌باز چگونه پروژه‌های متن‌باز را اجرا می‌کند و شما چگونه از روشهای آنها برای پروژه‌های خود استفاده می‌کنید. با فرض اینکه شما در حال توسعه سیستم‌عامل لینوکس هستید، می‌توانید تناظر یک به یکی بین جامعه متن‌باز و سازمان خود برقرار کنید و نکته اصلی همینجاست. مسلماً تفاوتهایی نیز وجود خواهد داشت اما نگذارید این تفاوتها شما را از به کارگیری این مزایا دور کند. در این روند، شما دید خوبی از نحوه توسعه نرم‌افزارها در این جامعه، بدست خواهید آورد.

این ساختار، درگیر مسائل مربوط به فروش، تولید و پشتیبانی نخواهد شد. در انتهای این فصل تأثیر آن را بر این گونه مسائل بررسی خواهیم کرد.



شکل ۹-۱: مدل عملیاتی توسعه متن‌باز بصورت شرکتی

معاونت مهندسی نرم‌افزار

فکر خود را مشغول پیچیدگیهای ساختار ارائه شده نکنید. در اینجا می‌خواهیم وظایف بالاترین مدیر مسئول در زمینه عرضه محصول به بازار را بررسی کنیم. وظیفه این معاونت این است که سیاستهای مربوط به محصول در حال تولید را طرح‌ریزی کرده، تولید آن را آغاز کند. مثلاً بداند که هسته لینوکس را طراحی می‌کند یا کارگزار آپاچی را؟ اغلب لازم است این پروژه را با سایر پروژه‌های در حال اجرا با سازمان مرتبط کرد. معاونت مهندسی نرم‌افزار باید ارتباط بین پروژه‌ها را بررسی و مستندسازی کند. با توجه به اینکه سازماندهی کارمندان در ساختار سازمانی ارائه شده مشخص است، او تمامی ابزار مورد نیاز را در اختیار دارد تا بتواند تشخیص دهد که آیا پیشرفت انجام شده هم‌جهت با سیاستهای سازمان است یا خیر. سؤالات اصلی که معاونت مدیریت نرم‌افزار باید به آنها پاسخ دهد عبارتند از:

- آیا ارتباط تیم بازاریابی با تیمهای مهندسی به طور مؤثر برقرار است؟ آیا بازار مورد نظر آنها مناسب است؟
- آیا گروه منابع انسانی توانسته است استعدادهای مورد نیاز را استخدام کند تا پروژه با موفقیت انجام شود؟
- آیا محصول در حال توسعه به درستی بر نیازهای بازار مورد نظر متمرکز شده است؟
- آیا درآمد حاصل از یک محصول منطبق با انتظارات سازمان است؟
- آیا خلاقیتها و نوآوریها، متناسب با نیاز بازار (یا سریعتر از آن) پیش می‌رود؟
- آیا تمامی اجزای مورد نیاز برای یک محصول عرضه شده‌اند؟
- آیا کیفیت محصول قابل قبول است؟ آیا سرعت واکنش ما به اشکالات محصور قابل قبول است؟
- آیا این محصول نیازهای کاربران خود را به خوبی برآورده می‌کند؟

- آیا نمودار نسبت هزینه به درآمد مطابق با انتظارات پیش می‌رود؟
- آیا دلگرمیهای سازمان منجر به بالا رفتن انرژی تیمها شده است؟
- آیا از بهترین استعدادهای موجود استفاده کرده‌ایم؟

اگر معاونت مهندسی نرم‌افزار این سؤالات را سرلوحهٔ خود قرار دهد، بهترین و خلاقانه‌ترین محصولات را با بالاترین کیفیت و سود تضمین شده به کاربران عرضه خواهید کرد. خوب بودن محصول خود باعث دلگرمی تیم و فعالیت بهتر آن می‌شود، زیرا اگر تیمی موفق شود، افتخار آن برای خود آنها خواهد ماند.

این معاونت وظیفه مهم دیگری نیز دارد. به زودی خواهید فهمید که معمار اولیه پروژه زمان آماده شدن محصول برای عرضه را اعلام خواهد کرد. در اکثر سازمانها فشار زیادی وارد می‌شود تا محصول زودتر عرضه شود اما در اینجا معاونت مهندسی نرم‌افزار **باید** صبر کند تا معمار اولیه تصمیم به عرضه محصول نرم‌افزاری گیرد. این کار، در روزهای اول پیاده‌سازی مدل بازار، بسیار مشکل خواهد بود. زیرا این مدل، با اکثر ساختارهای مدیریتی منافات دارد و باعث اعتراضاتی در سازمان خواهد شد. اما با ارائه اولین محصول به بازار این فشارها کاهش خواهد یافت، اما هیچ‌گاه متوقف نخواهد شد.

تیم فناوری

بر اساس ساختار، تیم فناوری از معمار اولیه، معمار عرضه محصول و مهندسان تشکیل شده است. این تیم بخش دیگری هم دارد که بعداً درباره آن صحبت خواهد شد و آن تیم «بِبَر به بازار» است. وظیفه تیم فناوری این است که اصل فناوری را تولید کند و وظیفه آماده‌سازی آن برای بازار را تیم «بِبر به بازار» برعهده دارد. فعلاً نقشهای اساسی تیم فناوری و برخی از خروجیهای آن را بررسی خواهیم کرد.

معمار اولیه

معمار اولیه باید ناظر اصلی باشد. این سمت تا حدی با مدیریت فناوری (CTO) در شرکتهای بزرگ فرق دارد. یک CTO در مقابل تصویر کلی فناوری یک شرکت مسئول و پاسخگوست. ولی معمار اولیه فقط تصویر فناوری محصول خاص را در نظر دارد. مثلاً با فرض پروژه پیاده‌سازی سیستم‌عامل لینوکس معمار اولیه باید بداند که هدف اصلی از این پروژه چیست: کارایی، قابلیت گسترش، توانایی بیشتر یا شبکه. اگر همه اینها اهداف پروژه هستند، اولویت‌بندی آنها چگونه است؟ چگونه می‌توان یک هدف را فدای دیگری کرد؟

برخی عقیده دارند که متخصصان فناوری قادر نیستند نیازهای مشتریان و الزامات بازار را درک و در سازمان اعمال کنند. اما جامعه متن‌باز اثبات کرده است که این ادعا اشتباه است. توجه شما را به این حقیقت جلب می‌کنم که یکی از مهمترین انگیزه‌ها برای هر مهندس نرم‌افزاری این است که بتواند محصول خود را به عموم عرضه کند. افرادی هم وجود دارند که مسخ زیباییهای فناوری می‌شوند. اما این افراد در اقلیت هستند. آنها که فریب زیباییهای فناوری را نمی‌خورند، معماران آینده شما خواهند شد.

باید معمار اولیه، خود را به کمک چارچوبهایی کلیدی هدایت کند تا تعادل خوبی بین نوآوری و بی‌رقیب بودن محصول برقرار شود. چارچوبهایی که شما باید متناسب با سازمان خود در نظر بگیرید عبارتند از:

- آیا محصول تولید شده قابل رقابت با هم نوعان خود در بازار می‌باشد؟
- آیا این محصول اهداف ارضاء کننده مشتریان (امکانات، کیفیت، کارایی) را برآورده می‌کند؟
- آیا فناوری در حال توسعه، خلاقانه است؟ آیا نقاط قوتی در این محصول وجود دارد که شما را از سایر تولیدکنندگان متمایز کند؟

این درست نیست که اجازه دهیم معمار اولیه به دلخواه خود جهت‌گیری کند. او نیاز به کمک دارد و اینجاست که تیمهای بازاریابی وارد می‌شوند. در کل باید حداکثر اطلاعات را در اختیار او قرار دهید تا بتواند بهترین تصمیم را بگیرد. یک معمار باسابقه، خودش برای بدست آوردن این اطلاعات تلاش خواهد کرد.

معمار اولیه باید مرز زیرسیستمها را نیز تعیین کند. هر پروژه نرم‌افزاری بزرگ، از زیرسیستمهایی تشکیل شده است. این زیرسیستمها به گونه‌ای باید تعیین شوند که حداقل وابستگی بین آنها وجود داشته باشد. از آنجایی که وابستگی به این زیرسیستمها اجتناب‌ناپذیر است، معمار باید واسط بین آنها را معین کند. با روشن شدن مرز بین سیستمها، معمار اولیه می‌تواند معماری هر یک از آنها را به شخص مورد اعتماد خود واگذار کند.

معمار اولیه همچنین در مقابل کدهای عرضه شده توسط معماران زیرسیستمها نیز مسئول است. از آنجایی که چارچوبهای مشخص شده، معمار اولیه را در مقابل محصول نهایی پاسخگو می‌داند، او مجبور است از معماران قوی برای معماری زیرسیستمها استفاده کند. مدیریت استعدادها از همین‌جا آغاز می‌شود بدین ترتیب که سلسله مراتبی از اعتماد بین افراد ایجاد می‌شود. اگر زیر سیستمی انتظارات را برآورده نکند، معمار جدیدی برای آن معرفی می‌شود. معمار اولیه باید چنان مقتدرانه عمل کند که بتواند به راحتی معمار جدید را جایگزین کند. اگر این انعطاف از دست برود، به همان مدل سنتی کاتدرال باز خواهید گشت. درجه حساسیت نسبت به کارهای ضعیف باید بسیار بالا باشد، یعنی از قبول چنین کارهایی به طور جدی جلوگیری شود. از وظایف معمار اولیه این است که با در نظر گرفتن پیامدهای عرضه محصول بی‌کیفیت، سعی کند محصولی بی‌رقیب، در زمان معین، تهیه کند.

معمار اولیه باید تشخیص دهد کِی زمان عرضه محصول فرا رسیده است. تشخیص وی باید بر اساس عواملی از قبیل کیفیت، امکانات و قابلیتهای محصول استوار باشد. در نهایت اوست که در مقابل پیامدهای آن مسئول شناخته می‌شود. اگر عرضه محصول قبل از موعد واقعی آن باشد، آنگاه

قابل رقابت نخواهد بود و اگر دیرتر از آن صورت گیرد، رقیبان بازار را از آن خود خواهند کرد. این عوامل متضاد، وی را به تشخیص بهترین زمان عرضه محصول راهنمایی خواهد کرد. شما به عنوان یک مدیر، باید پاداش‌ها و جریمه‌های مناسبی برای او در نظر بگیرید تا انگیزه کافی برای تصمیم‌گیری بهتر داشته باشید. این پاداش‌ها و جریمه‌ها نباید دروغین یا غیره واقعی باشند. در غیر این صورت نتیجه عکس خواهند داد. در عین حال نباید به گونه‌ای عمل کنید که گویا او تنها نابغه در این سیستم است. معمارها باید نابغه‌های تیم فناوری را از دیگران تمییز دهند و با جرأت از آنها بهره گیرند.

نکته آخر اینکه معمار اولیه باید رابطه دو طرفه‌ای را با معمار عرضه محصول، برقرار و حفظ کند. وقتی که محصول آماده شد، معمار عرضه وظیفه دارد که اصلاحات و بهبودهای جزئی را روی آن انجام دهد. از آنجایی که معمار عرضه در موقعیتی قرار دارد که تصمیم‌گیریهای او طراحی و معماری آینده کل سیستم را تحت تأثیر قرار می‌دهد، باید اعتماد دو طرفه‌ای بین او و معمار اولیه وجود داشته باشد. ارتباط این دو معمار شبیه رابطه‌ایست که لینوس و مارچلو هنگام طراحی هسته لینوکس با یکدیگر داشته‌اند.

معمار عرضه

معمار عرضه را باید معمار اولیه‌ای دانست که در حال انتظار است. همچنین می‌توان او را به عنوان «ستاد جبران حوادث ناخواسته» تلقی کرد. با توجه به اینکه ممکن است شرایطی پیش آید که مجبور به جایگزینی معمار اولیه با شخصی دارای مهارتهای مورد نیاز شوید، معمار عرضه می‌تواند جای خالی او را موقتاً پر کند و از بحران در روند پروژه جلوگیری کند. بسیاری این سؤال را پرسیده‌اند که «اگر لینوس توروالدز را اتوبوس زیر بگیرد چه بلایی سر لینوکس خواهد آمد؟» جامعه متن‌باز یک سیستم بازگشت به شرایط امن را تعبیه کرده است. در صورت چنین اتفاقی، مارچلو می‌تواند بلافاصله جایگزین لینوس شود و حرکت لینوکس را ادامه دهد. گرچه ممکن است در آینده جامعه متن‌باز شخص دیگری را برای راهبری لینوکس مناسب تشخیص دهد.

صرفنظر از شرایط فوق، بعد از اینکه معمار اولیه محصول را به سازمان واگذار می‌کند، فعالیت معمار عرضه شروع می‌شود. هر چند کیفیت محصول بالا باشد، باز هم اشکالات یا نقاط ضعفی در آن وجود خواهد داشت. در بعضی از سیستمها ممکن است خللهای امنیتی وجود داشته باشد که باید به سرعت برطرف شود. معمار عرضه مسئول پاسخگوئی به مشکلاتی است که بعد از عرضه نرم‌افزار آشکار می‌شود. او در عین حال نباید کلیت معماری سیستم را تغییر دهد. اگر پیشنهادهای برسد که اعمال آنها نیازمند تغییر اساسی در معماری سیستم باشد، در این صورت این پیشنهادها باید به معمار اولیه منتقل شود تا او در زمینه اعمال آنها در نسخه‌های بعدی تصمیم‌گیری کند.

همان گونه که در نمودار سازمانی می‌بینید، معمار عرضه نیز با معماران زیرسیستمها در تعامل است. نکته حائز اهمیت این است که باید نقاط کنترل مشترکی برای هر زیرسیستم وجود داشته باشد. در ادامه نقش معماران زیرسیستمها را بررسی خواهیم کرد. سپس در این زمینه بحث خواهیم کرد که چگونه معمار عرضه می‌تواند تیمها را تشویق به حل مشکلات پیش آمده در نرم‌افزار عرضه شده، کند.

به دلایل متعددی می‌توان گفت که وظایف معمار عرضه بسیار سنگین‌تر از معمار اولیه است. معمار عرضه باید به اندازه معمار اولیه، از معماری سیستم آگاهی داشته باشد و در عین حال باید به تغییرات بازار نیز واقف باشد. به طور خلاصه معمار عرضه کپی معمار اولیه است، با این تفاوت که حق ایجاد تغییر اساسی در معماری سیستم را ندارد.

چارچوب کاری معمار عرضه بر حفظ کیفیت و پایداری محصول استوار است. معمار عرضه باید بتواند در مدت کوتاهی برنامه‌های اصلاحی نرم‌افزار عرضه شده را آماده کند. در این صورت شما نگرانی از بابت کیفیت نخواهید داشت. اگر معمار عرضه اطلاعات درستی از کلیت سیستم نداشته باشد، در انجام وظیفه پاسخگویی به مشتریان به مشکل برمی‌خورد. از طرف دیگر اگر او نتواند رابطهٔ مورد نیاز را با تیمها برقرار کند، قادر نخواهد بود تغییرات مورد نیاز را به سیستم اعمال کند. با

نظرسنجی دربارهٔ کیفیت محصول می‌توان از میزان رضایت مشتریان و نحوهٔ عملکرد معمار عرضه مطلع شد.

معماران زیرسیستمها

معماران زیرسیستمها در حقیقت سلسله مراتبی از معماران یا طراحان سیستم هستند. عمق این سلسله مراتب بستگی به پیچیدگی سیستم دارد. در این مبحث عمق سلسله مراتب را یک لایه در نظر می‌گیریم. اگر لایه‌ها افزایش یابد، معمار سیستم بالادست نقش معمار اولیه را برای سیستم پایین دست ایفاء می‌کند.

اگر مروری به فناوری سیستم‌عامل لینوکس بیاندازید، به تعدادی زیرسیستم برمی‌خورید: حافظه مجازی، فایل سیستمها، مدیریت پروسه‌ها، شبکه و غیره. همان گونه که قبلاً هم اشاره شد مرز میان این زیر سیستمها توسط معمار اولیه تعیین می‌شود و ممکن است در روند تکامل سیستم، این مرزها تغییر کند. معمار سیستم حتی الامکان نباید پارامترهای تعیین شده برای زیرسیستم را تغییر دهد. اما می‌تواند بر تصمیم‌گیریهای معمار اولیه تأثیر گذارد.

معمار زیرسیستم در مقابل طراحی خودی زیر سیستم خود مسئول است. او کارهای انجام شده توسط مجموعه مهندسینِ تحت امر خود را دریافت و ارزیابی می‌کند. او باید با گروه منابع انسانی نیز در تماس باشد و از موجود بودن افراد با استعداد، در صورت نیاز احتمالی، اطمینان حاصل کند. این انگیزه در مهندسان وجود دارد که کارهای خود را به چندین زیرسیستم عرضه کنند تا بتوانند در سلسله مراتب ایجاد شده، ارتقاء یابند. اما این انگیزه را باید معماران سیستمها در آنها ایجاد کنند و این ضروریست.

وقتی که صحبت از ایجاد یک فناوری جدید می‌شود، معماران زیر سیستمها باید سیستمی با کیفیت و عملیاتی به معمار اولیه (یا معمار زیر سیستم بالادست خود) ارائه کنند. معمار اولیه باید چنان معماران زیر سیستمهای خود را قبول داشته باشد که نیازی به بازنگری کارهای آنها نداشته باشد.

یک معمار زیر سیستم وظایفی نیز نسبت به معمار عرضه دارد. هر اشکالی که در زیر سیستم وجود داشته باشد از طریق معمار عرضه دریافت می‌شود. معماران زیر سیستمها باید به تعامل با معمار عرضه تشویق شوند تا هر چه بهتر و سریعتر اصلاحات مورد نیاز را انجام دهند. هر چه واکنش به مشکلات بهتر صورت گیرد، کیفیت کل سیستم بالاتر می‌رود.

برخی از چارچوبهایی که شما می‌توانید برای معماران زیرسیستمها قرار دهید عبارتند از:

- **توانایی جذب استعدادها به سمت زیر سیستم خود**- به علاوه خود شما نیز باید از موجود بودن استعدادهای مورد نیاز اطمینان حاصل کنید.
- **تحویل خروجی با کیفیت و به موقع**- این مسأله را معمار زیرسیستم بالاتر باید تشخیص دهد. اگر خروجی زیر سیستمی با اهداف مورد نظر منطبق نباشد، معمار زیر سیستم باید عوض شود.
- **نوآوری در زیر سیستم**- فناوریهای جدید و ابداعات باید در کل سیستم صورت گیرد.
- **تعیین جانشین**- همان گونه که گفتیم معماران عرضه و اولیه، یک سیستم «جبران حوادث ناخواسته» تشکیل می‌دهند، اما معماران زیر سیستمها لزوماً چنین سیستمی ندارند. هرچند بهتر است چنین سیستمی را خودشان تعبیه کنند. در صورتی که معمار زیرسیستمی قصد داشته باشد در سلسله مراتب ارتقاء پیدا کند، باید جانشینی برای خود تعیین کرده باشد. بدین ترتیب سیستم جبران حادثه را نیز تشکیل داده است.

همان گونه که مشاهده می‌کنید، طی این فرآیندها شما یک فرهنگ جدید در سازمان خود ایجاد می‌کنید. به جای اینکه سلسله مراتب سازمان به گونه‌ای باشد که تصمیم‌گیریها از بالا به پایین صورت گیرند، فرهنگی تکامل می‌یابد که بر اساس رشد مهندسان و تقلای آنهای برای انجام دادن کار بهتر استوار است.

مهندسان

مهندسان سازمان شما، دانش شما هستند، آنها دارایی گرانبهای شما هستند و اجزای قابل تعویض نیستند. وقتی که مهندسی وارد سازمان می‌شود، نیاز به زمان دارد تا بتواند خود را با فرهنگ سازمان وفق دهد و سیستم در حال طراحی را بفهمد. افراد مختلف ممکن است اهداف متفاوتی داشته باشند. برخی می‌خواهد تا بالاترین سطح سلسله مراتب رشد کنند، برخی دیگر می‌خواهند با انجام دادن کار بیشتر دانش خود را گسترش دهند. عده‌ای نیز تمایل دارند در قلهٔ یک فناوری خاص قرار گیرند. شما باید ارزش هر یک از این سه گونه از مهندسان را تعیین کنید. همه آنها به ارزش سیستم می‌افزایند.

برای اینکه یک مهندس بتواند در سلسله مراتب ارتقاء یابد، باید بتواند علاوه بر استعداد فنی، تا حدی مهارت معماری خود را نیز نشان دهد. حتی مهارتهای روابط عمومی فرد نیز در این زمینه نقش بسزایی دارد. ارزش مهارتهای این افراد هر چه در سلسله مراتب بالاتر می‌روند، بیشتر می‌شود.

استعداد مهندسی افراد، باید از طریق کیفیت، کمیت، تنوع و نوآوری کارهای آنها سنجیده شود. اگر زیرسیستمی به نقطه خاصی از رشدیافتگی برسد، مهندسین آن زیرسیستم یا باید خلاقیت خود را در آن زیرسیستم نشان دهند، یا به زیرسیستمهای دیگر منتقل شوند. اگر کمیت کارها کاهش یابد، باید آن استعداد را با دلگرمی دادن احیا کرد.

معیار مهمی که می‌توان بر آن تکیه کرد، تعداد کارهای مردود مهندسان است. از آنجایی که تمامی سطوح سلسله مراتب در مقابل کیفیت کلی سیستم مسئول هستند، کارهای ضعیف و بی‌کیفیت با قاطعیت رد می‌شوند. برخی از کارهای مردود، طبیعی و قابل قبول‌اند. اما ارائه کارهای مردود به طور مکرر، نشانگر وجود مشکلی در میان مهندسان است که باید برطرف شود.

گروه منابع انسانی به طور پیوسته باید استعداد مهندسی سازمان را بهبود دهد. این گروه را با جزئیات بیشتری بررسی خواهیم کرد. دقت کنید که باکس مهندسان ضمائمی هم دارد. این ضمیمه‌ها در یک سازمان بزرگ نقش کلیدی دارند. یکی از مزایای مهم روش توسعه بازار این است

که همه می‌توانند وارد بازی شوند. در عین اینکه کارمندان شما باید از آموزش کافی در زمینه ساختار سازمان برخوردار باشند، فرهنگی که ایجاد می‌کنید نیز باید به گونه‌ای باشد که بتواند از خارج سازمان نیز نیروی کار جذب کند. اگر روزی دیدید که بخشهای دیگر سازمان نیز مایل به انجام کاری در زمینه بهبود سیستم هستند، بدانید که قدم بزرگی در رشد فرهنگ سازمان خود برداشته‌اید. چنین شرایطی اولاً علامت این است که سیستم کنونی پاسخگوی نیازهای کاربران است، ثانیاً علامت آن است که شما نیروهای مستعد و با انگیزه جدیدی در اختیار دارید که می‌توانند سیستم را بهبود دهند. کارهای این گروه نیز باید توسط یکی از معماران ارزیابی و سپس وارد سیستم شود. طبیعتاً کارهای ضعیف باید رد شوند. این باعث می‌شود سایر کارمندان شرکت نیز بدانند که اگرچه تشویق به کمک به بهبود سیستم می‌شود، اما فقط کارهای با کیفیتی که در چارچوبهای معماری سیستم می‌گنجند، مورد قبول واقع می‌شوند.

جای دیگری که ممکن است نیروی مستعد وجود داشته باشد، مشتریان و شرکاء شما هستند. این بخش در طبقه‌بندی **جوامع دروازه‌ای** می‌گنجد که در همین فصل بررسی خواهد شد.

منابع انسانی

جامعه متن‌باز اصولاً خود را زیاد درگیر هزینه‌های نیروی انسانی نمی‌کند. آنها لزومی نمی‌بینند که در زمینه پاداشها، کشیدن چک و تهیه تجهیزات نگرانی داشته باشند. گروه منابع انسانی، از موارد اصلی اختلاف بین روش توسعه بازار در جامعه و شرکتهاست.

نقشها و وظایف گروه منابع انسانی که در اینجا شرح داده خواهد شد، ممکن است فراتر از آنچه باشد که شما از آنها انتظار دارید. نقش این تیم بسیار حساس است. هدف گروه فناوری این است که تمام انرژی خود را بر فناوری و آماده کردن محصول متمرکز کند. بدین ترتیب مدیریت درست کارمندان در این راستا بسیار مهم است. این وظیفه را یا باید خود تیم بر عهده گیرد یا به تیم منابع انسانی واگذار کند.

رشد استعدادها

نیروی انسانی برای رشد استعدادهای سازمان باید منابع مورد نیاز را در اختیار داشته باشد. ممکن است لازم باشد کارمندان از دوره‌های آموزشی خارج سازمان استفاده کنند یا در یک ناحیه خاصی کارآموزی کند. در شرکتهای سنتی رشد افراد بر عهده مدیر مستقیم آنها می‌باشد. در این روش مدیریت مستقیم دیگر وجود ندارد، اما همچنان منابعی برای رشد افراد مورد نیاز است.

ایجاد ارتباطات داخل سازمانی نیز، برای رشد کارمندان لازم است. همچنین مدیریت شرکت باید فرآیندی را برای انتقال سیاست‌گذاریهای شرکت به کارمندان در نظر گیرد. بدین ترتیب علاوه بر اینکه کار روزانه گروه مهندسی شرکت منطبق با اهداف فناوری در حال توسعه می‌گردد، ارتباطات داخل شرکت باعث می‌شود کارمندان از اهداف استراتژیک شرکت نیز آگاهی یابند.

مدیریت مردم (مهارتها)

این گروه وظیفه دارد که نیازهای مهارتی معماران را برآورده کند و از موجود بودن نیروی ماهر اطمینان حاصل کند. اگر مهارت کمیابی مورد نیاز باشد، باید در جدب این مهارت سرمایه‌گذاری خوبی شود. از وظایف دیگر این تیم این است که حتی‌الامکان طیف گسترده‌ای از انواع مهارتها را آماده به کار داشته باشد. منظور از طیف گسترده مهارتها این است مخلوطی از افراد ماهر علاقمند به فناوری خاص است که در سلسله مراتب مدیریتی رشد می‌کنند و به درجات مختلفی از مدیریت می‌رسند.

این تیم باید توانایی مالی شرکت را نیز در استخدام کارمندان مدنظر داشته باشد. این کار را می‌توان از طریق استخدام کارمندان قراردادی و دائم محقق کرد. اما اگر شرکت قادر نباشد، نیروی جدیدی استخدام کند. وظیفه این تیم است که با گروه‌های مهندسی مذاکره کند و نیروهای ماهرتر را به زیرسیستمی که نیاز بیشتری دارد، منتقل کند. در این شرایط ممکن است مهندسان به این نتیجه برسند که فعالیت خود را بر روی یک زیرسیستم متوقف کنند و بر زیرسیستم مهمتری

تمرکز کنند. اما عده‌ای هم ممکن است تصمیم به بیکار ماندن تا باز شدن مجدد آن زیرسیستم بگیرند که عواقب منفی آن متوجه خودشان خواهد شد.

قسمتهای دیگر مدیریت مهارت باید از موجود بودن ماهرترین افراد و رشد آنها اطمینان حاصل کند. این تیم باید فرآیندهایی را برای محک زدن کار مهندسان تعبیه کند. معیارهای سنجش کار مهندسان عبارتند از: سرعت، تنوع و میزان کارهای مردود. آنها باید با کم‌کاری‌های کارمندان، به طور جدی برخورد کنند. گرچه این روش ممکن است خشن به نظر برسد، اما باعث ایجاد سیستم خود اصلاحی می‌شود و افرادی که نمی‌توانند خوب کار کنند، خودشان باید دنبال راه حل دیگری داخل یا خارج شرکت باشند. اینکه چه مدت لازم است تا فردی بتواند کار مفید و پیوسته ارائه دهد (با شرایط سازگار شود) بستگی به فرهنگ جاافتاده در شرکت دارد.

آموزش

این تیم وظیفه آموزش کلی کارمندان و آموزش فرآیندی مهندسان را بر عهده دارد. آموزش کلی، اغلب ترکیبی است از آموزش فرهنگ کار و شرح اهداف و سیاستهای شرکت. آموزش فرآیندی مهندسان نیروهای ماهر را آماده کار در زیرسیستمها می‌کند. در این آموزشها می‌توان فناوری در حال توسعه را تا سطوح پیشرفته آموزش داد، اما تمرکز اصلی باید بر ابزار مورد استفاده مهندسان و نحوه انجام کار باشد. اکثر مهندسان مایلند هر چه زودتر آموخته‌هایشان را در عمل اجرا کنند. آنها می‌توانند پس از یادگیری فرآیند کار، با تمرین و تعامل با سایر مهندسان، کار خود را شروع کنند. نقش تیم آموزش را دست کم نگیرید. اگر مهندسان از نحوه مؤثر انجام کار مطلع نباشند، همه ضرر خواهند کرد.

آزمایش و یکپارچگی

این هم از دیگر وظایف مربوط به شرکت‌ها است که در جامعه متن‌باز به چشم نمی‌خورد. فرآیند آزمایش کردن و یکپارچه کردن نرم‌افزارها در جامعه متن‌باز به سادگی ایمیل کردن فایلهای

منبع، وصله‌ها و اصلاحات به نگهدارندهٔ آن نرم‌افزار می‌باشد. مجموعه ابزاری که در فصل ۳ به آنها اشاره شد (CVS، Bugzilla و ...) نقاط اتکای مهمی برای این جامعه هستند. آزمایش کردن نرم‌افزار به این ترتیب انجام می‌شود که جمعیت عظیمی از کاربران در جاهای مختلف دنیا، حین کار با آن نرم‌افزار، اشکالات را پیدا می‌کنند. در ناسا به ازاء هر برنامه‌نویس از ۳۰ مهندس، برای آزمایش برنامه‌ها استفاده می‌شود. در حالی که در جامعه متن‌باز، نرم‌افزارها رسماً آزمایش نمی‌شوند. سؤال اصلی این است که با این وجود چگونه جامعه متن‌باز توانسته است به نتایج کیفی قابل توجهی دست یابد. ضرب‌المثل ریموند را به خاطر آورید که «اگر انسان به اندازه کافی چشم داشت، همه حشرات (Bug) ناتوان می‌شدند.» این است راز کیفیت نرم‌افزارهای متن‌باز.

در اغلب شرکت‌ها دسترسی به کد منبع به شدت کنترل می‌شود و فقط مراجع خاصی (مثلاً دولت‌ها) قادر به نگاه کردن به آن هستند. همچنین تغییر دادن در کد منبع نیز برای عدهٔ اندکی امکان‌پذیر است. شما باید این فرهنگ را نهادینه کنید که هر چه افراد بیشتری کد منبع را ببینند، بهتر است. نظرات در زمینه طراحی، نحوه کدنویسی و الگوریتم‌های برنامه را باید به گرمی بپذیرید، اما بخاطر داشته باشید که در نهایت شما می‌خواهید محصول را به صورت تجاری عرضه کنید. بنابراین کاربران شما برای تست کردن و کنترل کیفیت محصول روی شما حساب می‌کنند و از آنجایی که نسبت برنامه‌نویسان به کاربران در مورد محصول شما بسیار کمتر از موردهای طراحی شده در جامعه متن‌باز است، لازم است محصول خود را به دقت آزمایش کنید.

در فصل دوازدهم در زمینه فرآیند در دسترس قرار دادن کد منبع، تحت یک مجوز متن‌باز بحث خواهیم کرد. معمولاً پیش‌بینی زمان این کار امکانپذیر نیست. بهتر است برای یکپارچه‌سازی کد منبع و انتقال آن به جامعه متن‌باز از همان ابزاری استفاده کنید که آنها استفاده می‌کنند. در برخی موارد قوانین و محدودیتهای شرکت، ممکن است شما را مجبور به استفاده از سیستمهای یکپارچه‌سازی خاصی کند. در چنین شرایطی باید به شرکتهایی مراجعه کنید که ابزارهایی برای ترکیب متن‌باز با روشهای سنتی ارائه کرده‌اند.

با این همه شما باید ساختار تست محصول را حفظ کنید. اگر فرهنگ جدید درست پیاده شده باشد، باید شاهد کاهش قابل توجه مشکلات یافت شده در هنگام تست محصول باشید. مدیران باتجربه باید بدانند که هدف از ایجاد تیم تست محصول فقط این نیست که اشکالات را آشکار کند، بلکه کیفیت کار کارمندان را نیز منعکس می‌کند.

گروه مالی

همکاران امور مالی شما نیز همان وظیفه سنتی خود را بر عهده خواهند داشت، اما با چند تفاوت جزئی. وقتی که افراد دارای مهارتهای مختلف را استخدام می‌کنید پایه سیستم دقیق برای حقوق و دستمزد آنها تعبیه کنید. گروه مالی باید نیازهای تیمهای فناوری و «بِبَر به بازار» را نیز در تعادل نگه دارد و در عین حال باید با گروه منابع انسانی نیز در ارتباط باشد تا هزینه نیروی انسانی از مقادیر پیش‌بینی شده فراتر نرود.

بازاریابی

از دیگر وظایفی که در جامعه متن‌باز به چشم نمی‌خورد بازاریابی است. به یادآورید که گفتیم در جامعه متن‌باز نویسندهٔ یک برنامه، استفاده کننده از آن نیز هست و این بسیار جالب است. اما وقتی که شما محصولی را برای داخل شرکت یا عرضه تجاری تهیه می‌کنید باید دید جامع‌تری از نیاز کاربران داشته باشید. در روشهای بازاریابی رایج، ابتدا مشتریان دسته‌بندی و اولویت‌بندی می‌شوند و سپس نیازهای آنها به ترتیب اولویت به عنوان اهداف شرکت در نظر گرفته می‌شود. بنابراین شما می‌توانید به روشنی بازارهای هدف کنونی و آیندهٔ خود را برای معمار اولیه مشخص کنید. این اطلاعات راهنمای بسیار خوبی در راستای پیشرفت پروژه هستند و این مزیت کار شرکتی نسبت به کار انجام شده جامعهٔ متن‌باز است و سود قابل توجهی عاید شرکت خواهد کرد.

وظیفه دیگر این گروه این است که چارچوبهای کاری تیمها را تعیین و به طور پیوسته با توجه به شرایط بازار به روز کند و بدین ترتیب جهت‌گیری آنها را منطبق بر نیازهای کاربران حفظ کند.

سنجش توانایی رقابتی محصول معمار خوبی از کیفیت کار معمار اولیه است. این گروه تأثیر جانبی دیگری نیز دارد. اکثر تیمهای بازاریابی مایلند مشخصات محصول را تا جزئیات ریز آن تعیین کنند. بنابراین تیمهای مهندسی احساس می‌کنند که جایی برای نوآوری و خلاقیت آنها در نظر گرفته‌اند. اما این تیم فقط بازارهای هدف و اهداف رقابتی مفید را معین می‌کند و مهندسان انگیزهٔ زیادی برای تهیهٔ فناوری مطلوب با امکانات مناسب خواهند داشت.

بِبَر به بازار

من تا اینجا تلاش کردم عرضه فناوری را به گروه فناوری ربط ندهم. علت این است که وظیفهٔ عرضه یک محصول شُسته رُفته بر عهده «تیم بِبَر به بازار» است. برای درک این مسئله کافیست نگاهی به نحوه عرضه سیستم‌عامل لینوکس بیاندازیم. لینوس توروالدز به همراه تیم ماهری از برنامه‌نویسان، کرنل لینوکس را تهیه می‌کنند. هزاران برنامه‌نویس دیگر نرم‌افزارهای سیستم GNU را تهیه می‌کنند و بهبود می‌دهند. سپس تعدادی از توزیع‌کنندگان لینوکس همه این اجزاء را با یکدیگر ترکیب می‌کنند و آنها را به صورت یکپارچه تحت یک محصول برای استفاده کاربران عرضه می‌کنند. آنچه که باید به آن اشاره شود این است که هر توزیع‌کننده‌ای، این اجزاء را به صورت خاصی جمع می‌کند که با نحوهٔ جمع‌آوری توزیع‌کنندهٔ دیگر متفاوت است. آنها ابزارها و قابلیتهایی را نیز به آن اضافه می‌کنند بسته به اینکه هدف آنها چه بازاری باشد. همان گونه که قبلاً اشاره شد برخی از توزیع‌کنندگان ممکن است بازار کارگزارها را هدف گرفته باشند و برخی دیگر بازار کامپیوترهای رومیزی را. هدف برخی دیگر از توزیع‌کنندگان برنامه‌های ارتباطی بلادرنگ می‌باشد.

این گروه نقش توزیع‌کننده را برای محصولات شرکت بازی می‌کند و می‌تواند کاری کند که شما بتوانید یک فناوری را به بخشهای مختلف بازار عرضه کنید. اگر پروژه‌های شما کوچک باشند، ممکن است یک تیم بِبَر به بازار نیاز شما را برآورده کند. اما در زمینه پروژه‌های بزرگ مسلماً یک تیم کافی نخواهد بود. مدیریت شرکت باید مرز بین پایان فعالیت گروه فناوری و آغاز فعالیت گروه بازار را دقیقاً مشخص کند. مفهوم این جمله این است که شما در گروه بِبَر به بازار نیز تا حدی

نیازمند مهارت مهندسی هستید. این مهندسان نیازمند مهارتهایی غیر از مهارتهای مورد نیاز در تیم مهندسی، از قبیل درک مسائل بازار، هستند.

وظیفه دیگر این تیم این است که زنجیره‌های ارزش محصول را کامل کند از قبیل تعامل با گروه‌های فروش، تولید، پشتیبانی و همهٔ گروه‌هایی که در عرضه یک محصول قابل استفاده، چه برای داخل سازمان و چه برای عرضه به مشتریان، نقش دارند.

سایر اجزای ساختاری

این مبحث بیشتر بر نحوهٔ انجام امور مهندسی و ارتباطات آن با سایر تیمها متمرکز شده است. تأثیر سیستم جدید بر سایر ارتباطات ساختاری نیز باید بررسی شود. این نکته حائز اهمیت است که بتوان تغییرات صورت گرفته در سازمان ناشی از روش مدیریت جدید را به گروه پشتیبانی، تولید، فروش و سایر بخشهای کلیدی سازمان منتقل کرد. به عنوان مثال گروه پشتیبانی از اینکه می‌تواند در توسعه و پیشرفت پروژه نقش داشته باشد، استقبال خواهد کرد و ممکن است ایده‌های خوبی نیز در این زمینه ارائه دهد. توسعه از بخشهای دیگری نیز تشکیل شده است که متناسب با شرایط سازمان، باید به آنها بپردازد. به عنوان مثال ممکن است شما برای اینکه از میزان کارایی محصول خود آگاه شوید یک تیم عوامل انسانی نیز به ساختار سازمانی اضافه کنید که با گروه مهندسی در تعامل باشد. به طور خلاصه نکته کلیدی در این روش، برقراری و حفظ ارتباط مؤثر بین اجزاء سازمان است.

جوامع دروازه‌ای

همچنان که فرهنگ سازمان شما رشد می‌کند و بازیگران بیشتری به جامعهٔ توسعه دهندگان شما اضافه می‌شود، به نقطه‌ای خواهید رسید که مشتریان و شرکای کلیدی شما نیز علاقمند خواهند شد که به فرآیند توسعه دخیل گردند. از آنجایی که نمی‌توان همه را برای شرکت در این فرآیند راه داد، باید این همکاری را به صورتی قانونمند درآورید.

فراموش نکنید که هدف این فصل این است که فرآیند توسعه بازار را در سازمان پیاده کنید، نه اینکه کد منبع خود را در اختیار عموم قرار دهید. جوامع دروازه‌ای به فرآیند دخیل کردن مشتریان و شرکا در امر توسعه در عین حفظ کردن کد منبع به صورت اختصاصی گفته می‌شود.

چه دلیلی دارد که مشتریان و شرکا را در فرآیند توسعه دخیل کنیم؟ یکی از مزایای اصلی متن‌باز واکنش سریع به مشکلات است. در دنیای نرم‌افزارهای اختصاصی، مشتریان بزرگ وقتی که شرکت‌های تولیدکننده نرم‌افزار نمی‌توانند خواسته‌ها و نیازهای آنها را برآورده کنند، ناامید می‌شوند. برخی از این خواسته‌ها در حد رفع یک اشکال جزئی‌ست و برخی دیگر ممکن است در حد افزودن امکانات پیچیده به نرم‌افزار باشد. شرکا به شرکتهایی اطلاق می‌شود که محصولشان متکی به محصول شماست. دخیل شدن شرکا نیز باعث بهبود کیفیت محصول شما خواهد شد. به همین دلیل بزرگترین بهره‌ای که از جوامع دروازه‌ای می‌برید، نوآوریهائیست که نیازهای مشتریان شما را برآورده می‌کند و پاسخگوئی به نیازهای مشتریان را سریعاً بهبود می‌بخشد. ایجاد جوامع دروازه‌ای نیازمند تشکیل زیرساخت شبکه‌ایست که به افراد خارج از سازمان اجازه دسترسی به کدهای منبع را بدهد و در عین حال امنیت کار شما را نیز تأمین کند. همچنین باید کنترلهایی نیز برای شناسائی شرکای اصلی که حق دخالت در پروژه را دارند، وجود داشته باشد.

شما باید قراردادهایی نیز در زمینه حفظ محرمانگی اسناد با شرکا منعقد کنید. اگر ساختار کد منبع را بر اساس قواعد متن باز بنا کنید، متوجه خواهید شد که شرکا خواهند توانست به طور مؤثری با جامعه برنامه‌نویسان پروژه شما همکاری کنند. همچنین باید توافق نامه‌ای در زمینه مالکیت معنوی کارهایی که از طرف مشتریان و شرکا ارائه می‌شود، تنظیم کنید. اگر کدهای منبع را تحت یکی از مجوزهای متن باز عرضه کنید، در این صورت کمترین محدودیتها را برای آن قائل شده‌اید.

خطرات و مشکلات

هر ساختار جدیدی با خطرات و نقصهایی دست و پنجه نرم می‌کند و ساختار بازار نیز از این قاعده مستثنی نیست. در ابتدای امر شما باید این ساختار عملیاتی را به یک ساختار سازمانی خاص سازمان خود تبدیل کنید. انتخاب بازیگران اصلی این ساختار تأثیر بسزایی بر موفقیت این روش دارد. در فصل سیزده فرآیند انتخاب افراد و چارچوبها را با جزئیات بیشتری بررسی خواهیم کرد. انتقال از یک ساختار قدیمی به یک ساختار جدید نیز خود خطراتی در پی دارد. اگر نقشه مهاجرت دقیق و با در نظر گرفتن جزئیات تهیه کرده باشید که ارتباطات نیز در آن در نظر گرفته شده باشد، در این صورت می‌توانید بر بسیاری از خطرات غلبه کنید. همچنین زمان پیاده‌سازی روش توسعه جدید در چرخهٔ عرضه محصول نیز باید با دقت تعیین شده باشد. همان گونه که چندین بار اشاره کرده‌ام، شما باید یک فرهنگ متن‌باز را داخل سازمان پیاده کنید که کارمندان کمترین محدودیت را برای دسترسی به کدهای منبع داشته باشند.

احتمالاً بزرگترین خطر مربوط، به نحوه مدیریت کارمندان می‌شود. از آنجایی که این ساختار نقش مدیریت مستقیم بر گروه‌های مهندسی را کم رنگ‌تر می‌کند، باید از مشاوران و مربیان به طور مؤثرتری استفاده کنید و گروه منابع انسانی نیز باید ارتقاء سطح مهارت مهندسان را بالاترین اولویت خود قرار دهد.

جمع‌بندی

در این فصل پیش زمینه‌های گسترده‌ای در زمینه موضوعی بسیار پیچیده در اختیار شما قرار گرفت. ممکن است حتی لازم باشد این فصل را چندین بار مطالعه کنید. در این فصل ما نحوهٔ عملکرد جامعه متن‌باز در یک دنیای تجاری را بررسی کردیم. امیدواریم شما علاوه بر درک نحوه رسیدن جامعه متن‌باز به اهداف خود، توانسته باشید برخی از روشهای آنها را در سازمان خود به کار گیرید.

قبل از اینکه دانش شما را در زمینه توسعه متن‌باز به عمومی کردن کد منبع گسترش دهیم، میانبری می‌زنیم به تأثیرات متن‌باز بر مدلهای تجاری رایج. این بحث دلایل متن‌باز کردن برخی از پروژه‌ها و دخیل کردن جامعه متن‌باز در محصولات و پروژه‌ها را روشن خواهد کرد.

فصل ۱۰

ارزش محصول به عنوان تابعی از زمان

فروشندگان نرم‌افزار به این امید تولید فناوری می‌کنند که درآمد فروش آن بیشتر از هزینهٔ تولید آن گردد. هر چه نرم‌افزار منحصر به فرد و ارزش آن بیشتر باشد، قدرت قیمت‌گذاری فروشنده بیشتر می‌شود. عوامل اولیهٔ تأثیرگذار بر قدرت قیمت‌گذاری محصول عبارتند از: ارزشی که خریدار برای محصول قائل می‌شود و فشار رقابتی موجود در بازار. متن‌باز با روشهای دیگری، قدرت قیمت‌گذاری را تحت الشعاع قرار می‌دهد و ارزش برخی از فناوریها را سریعتر پایین می‌آورد. در این فصل روشهای جدید رقابت در این بازار بررسی می‌شود. گرچه این فصل برای فروشندگان نرم‌افزار به صورت تجاری نوشته شده است، اما برای مدیران فناوری اطلاعات نیز از این جهت مفید است که عوامل تأثیرگذار بر فروشندگان نرم‌افزار را درک می‌کند و دید بهتری نسبت به روشهای رقابتی جدید در حال گسترش در بازار پیدا می‌کنند. درک نحوهٔ تأثیرگذاری متن‌باز بر منابع سازمان نیز برای مدیران فناوری اطلاعات حائز اهمیت است. اهداف این فصل این است که دید بهتری نسبت به موارد زیر پیدا کنید:

- تأثیر اقتصادی منحنی ارزش- زمان
- تأثیرات متن‌باز بر ارزش نرم‌افزارهای تجاری
- تأثیر متن‌باز بر حفظ ارزش نرم‌افزار
- چگونه از تأثیرات متن‌باز بر منحنی ارزش- زمان بهره‌برداری شود

به عنوان یک تولیدکننده نرم‌افزار تجاری، شما باید نرم‌افزاری تولید کنید که ارزش آن بالاتر از ارزشمندترین نرم‌افزار موجود در بازار باشد، در غیر این صورت درآمد شما در حدی نخواهد بود که بازگشت سرمایه را تضمین کند. برای اینکه به درک بهتری از تأثیرات نرم‌افزارهای متن‌باز بر کسب و کار برسید نگاهی می‌اندازیم به تحولات بازار یکی از صنایعی که مدتی تحت تأثیر آن بوده است.

صنعت داروسازی

صنعت داروسازی سالهاست که درگیر مشکل کاهش ارزش محصولات پس از طی زمانی مشخص می‌باشد. در اواسط دهۀ ۱۹۸۰ من از درد معده رنج می‌بردم بعد از مدت حدود ۱۸ ماه این درد چنان شده بود که پس از هر بار غذا خوردن مرا آزار می‌داد. دکترهای زیادی مرا معاینه کردند، گرچه در آن زمان معالجۀ قطعی برای این مشکل وجود نداشت، اما می‌شد درد آن را با مصرف برخی داروها خفیف‌تر کرد. متأسفانه بدن من نسبت به این دارو واکنش نشان می‌داد و قادر به مصرف آن نبودم. در همان زمان داروی جدیدی به نام «رانیتیدین» به بازار آمد که امروزه به نام «زانتاک» نیز شناخته می‌شود. بعد از مصرف این داروی جدید به مدت یک هفته تمامی علائم بیماری من برطرف شد و به ۱۸ ماه درد کشیدن من پایان داده شد. من حاضر بودم این دارو را به هر قیمتی تهیه کنم. از نظر من این دارو یک معجزه بود. خوشبختانه من تحت پوشش بیمه خوبی بودم. هر روز باید ۱۲۰۰ میلی‌گرم از این دارو مصرف می‌کردم. با توجه به اینکه قیمت هر قرص ۱۵۰ میلی‌گرمی ۱/۸۰ دلار بود، مصرف روزانه من ۱۵ دلار در روز یا ۵ هزار دلار در سال می‌شد. از آنجایی که مصرف هر داروی دیگری وضعیت مرا بدتر می‌کرد، شرکت بیمه کننده چاره‌ای جز پرداخت این هزینه نداشت. در آن زمان هیچ رقابتی در زمینه رانیتیدین وجود نداشت. من به مدت هفت سال این دارو را مصرف کردم و باید اعتراف کنم که اگر سیگار نمی‌کشیدم، مدت درمان کوتاه‌تر می‌شد.

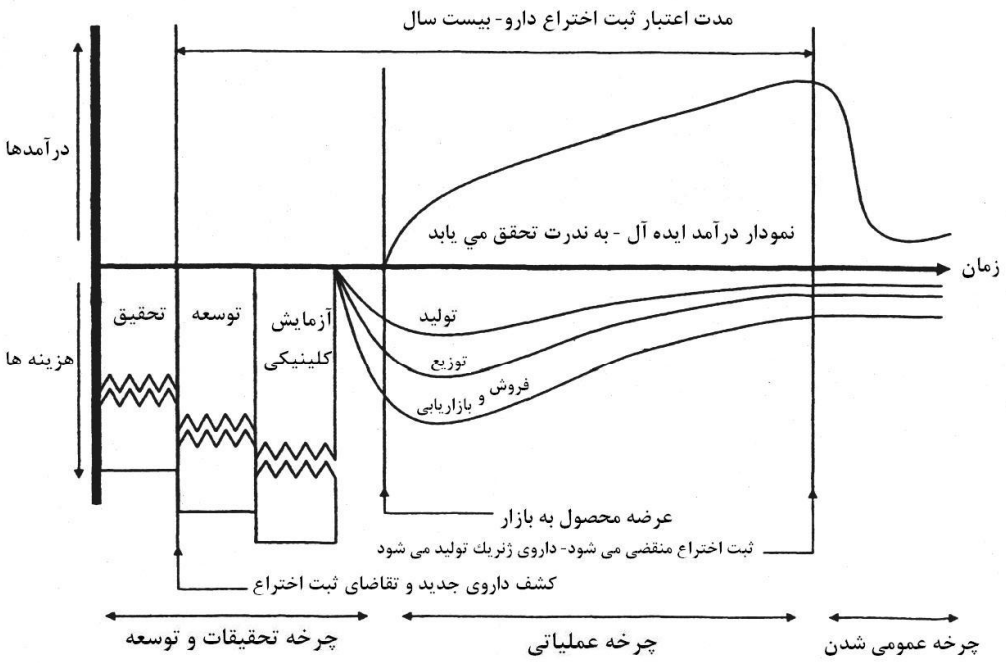
اخیراً قیمت رانیتیدین را پرسیدم و متوجه شدم که قیمت هر قرص ۱۵۰ میلی‌گرمی آن کمتر از ۱۲ صدم دلار است. درمان هفت ساله ۳۵ هزار دلاری من امروزه کمتر از ۲۵۰۰ دلار می‌شد. یعنی هزینه درمان من ۹۳ درصد کاهش یافته است. آیا می‌توانید تصور کنید که محصول شما به این سرعت افت قیمت پیدا کند. علت کاهش قیمت شدید رانیتیدین این بود که نوع ژنریک آن به بازار آمد. داروهای ژرنیک پس از اینکه حق ثبت تولیدکننده اصلی آن منقضی می‌شود، به بازار می‌آید. در مورد داروهای مفیدتر نسخۀ ژنریک آنها در همان روز انقضاء حق ثبت عرضه می‌شود.

اکنون باید ببینیم صنعت داروسازی چگونه بر هزینه‌های خود فائق آمده است و مدل تجاری خود را چگونه تنظیم کرده است. سپس به صنعت نرم‌افزار باز می‌گردیم و تجربهٔ صنعت داروسازی را در آنجا اعمال می‌کنیم.

هزینه، ارزش، بازگشت سرمایه و زمان

صنعت داروسازی از این واقعیت بهره می‌برد که زمان دقیق انقضاء حق ثبت یک دارو مشخص است و همچنین پیامد آن که عرضه داروهای ژنریک به بازار می‌باشد، واضح است. این مسئله به صنعت داروسازی این امکان را می‌دهد که مدل هزینه به درآمد را دقیقاً بنا کند.

نگاهی به نمودار شکل ۱۰-۱ می‌اندازیم.



شکل ۱۰-۱ مدل بازگشت سرمایه تولیدات دارویی

در این نمودار چرخهٔ اقتصادی داروهای جدید به زبان ساده نمایش داده شده است. من قصد ندارم شما را متخصص مدلهای اقتصادی این صنعت کنم، بلکه می‌خواهم اثرات آن را با اثرات متن‌باز بر صنعت نرم‌افزار مشابه‌سازی کنم.

شرکتهای داروسازی هزینهٔ زیادی را صرف تحقیقات برای تولید داروی جدید می‌کنند. هدف آنها این است که روشهای درمانی جدیدتری برای بیماریهای بی‌شمار جامعهٔ بشری کشف کنند.

انگیزهٔ اصلی این شرکتها برای صرف هزینه‌های سنگین تحقیقاتی این است که کلیهٔ حقوق اکتشافات آنها، در صورتی که مورد قبول ادارهٔ ثبت اختراع واقع شود، به مدت ۲۰ سال در انحصار آنها خواهد بود. در مواردی نیز می‌توان آن را تمدید کرد.

شکل ۱۰-۱ مدل اقتصادی بازگشت سرمایه برای داروهای کلینیکی

در نقطه‌ای از چرخهٔ تحقیق، داروی جدید کشف می‌شود. شرکت برای جلب حمایت از این کشف درخواست ثبت اختراع را می‌کند. حدود ۲ تا ۴ سال طول می‌کشد تا حق ثبت به آنها داده شود. مرحلهٔ بعد این است که کشف جدید وارد فرآیندهای توسعه شود تا قابل عرضه به بازار گردد. فرآیندهای توسعه شامل تصفیه دارو، تست آن بر روی موجودات غیرانسانی و ... می‌شود.

چرخهٔ توسعه با آزمایشهای کلینیکی تکامل می‌یابد. در این مرحله دارو به صورت کنترل شده بر روی انسان‌ها تست می‌شود. در این تست‌ها سلامت، میزان تأثیرگذاری و دوز مناسب آن مشخص می‌شود. اگر از سلامت دارو اطمینان حاصل گردد، چرخهٔ تولید آن قبل از گرفتن تأئیدیه از سازمانهای ذیربط، آغاز می‌شود. اما قبل از تکمیل آزمایشهای کلینیکی و تأئیدیه سازمانهای ذیربط، این دارو قابل عرضه در بازار نیست. در برخی موارد ممکن است دارو تأئید نشود که در این صورت باید فرآیند تولید متوقف گردد. در نظر داشته باشید که به محض رسیدن دارو به بازار شمارش معکوس برای انقضای حق ثبت آغاز می‌شود. شرکت تولیدکننده دارو تلاش می‌کند که توسعه و آزمایش محصول و گرفتن تأییدیه آن را در کمترین زمان ممکن انجام دهد و در عین حال محصولی سالم عرضه کند.

این شرکتها قبل از گرفتن تأییدیه اقدامات دیگری نیز انجام می‌دهند از قبیل آماده کردن تیم تولید، آموزش تیم فروش و آماده کردن بازار برای محصول فوق‌العاده و جدید خود. سپس چرخهٔ عملیاتی آغاز می‌گردد. در این چرخه هدف این است که درآمد ناشی از این محصول به حداکثر برسد. این محصول به تنهایی نه فقط هزینه‌های تولید، فروش، بازاریابی و توزیع بلکه هزینهٔ سنگین

چرخه‌های تحقیق و توسعه را نیز باید پوشش دهد. اگر درآمد فروش دارو به نقطه‌ای برسد که بتواند هزینه‌های عملیاتی و تحقیق و توسعه را پوشش دهد، بازگشت سرمایه آن مثبت می‌شود.

یکی از پیچیدگیهایی که در این نمودار نمایش داده نشده است، این است که در هر زمانی ممکن است تحقیقاتی بر روی محصول دیگری در حال انجام باشد و برخی از این تحقیقات ممکن است هیچ‌گاه نتیجه ندهد. آمار نشان می‌دهد که بسیاری از داروهای جدید نمی‌توانند به بازار راه یابند بنابراین داروهایی که وارد بازار می‌شوند، باید علاوه بر پوشش هزینه‌های چرخه‌های عملیاتی و تحقیق و توسعه خود، هزینه‌های تحقیقات شکست خورده را نیز جبران کنند. از طرف دیگر درآمد این داروها باید چنان بر نقدینگی شرکت بیافزاید که علاوه بر صرف قسمتی از آن برای تحقیقات جدید، سودی نیز عاید سرمایه‌گذاران کند.

همان گونه که قبلاً گفته شد یکی از برتریهای صنعت داروسازی این است که دقیقاً می‌داند چه زمانی حق ثبت دارو منقضی می‌شود. بنابراین شرکتها می‌دانند که چه زمانی ارزش محصولشان به واسطهٔ آمدن داروهای ژنریک کاهش می‌یابد. بدین ترتیب در اینجا می‌توان مدل هزینه‌ای بسیار دقیقی طراحی کرد. به کمک آن می‌توان قیمتی برای محصول در نظر گرفت که با وجود همهٔ هزینه‌های عملیاتی، بازگشت سرمایه و سود خوبی را تضمین کند و در عین حال برای بازار قابل تحمل باشد. از آنجایی که زمان عرضه داروی ژنریک مشخص است، شرکت تولید کننده می‌تواند به موقع، اقدام به کاهش هزینه‌های عملیاتی کند. به طور کلی قابل پیش‌بینی بودن وقایع بازار طراحی مدل هزینه‌ای بسیار دقیق و مؤثری را امکانپذیر می‌کند، حتی اگر میزان موفقیت تحقیقات مشخص نباشد.

بازیابی از شرایط عمومی شدن محصول

تصور کنید چه اتفاقی می‌افتد وقتی که داروهای ژنریک در دسترس عموم قرار گیرد و بازار دیگر ارزش بالایی برای محصول شما قائل نباشد؟ در این شرایط رقابتی جدید، شرکت باید طرح

تجاری جدیدی برای خود طراحی کند تا درآمد حاصل از فروش آن دارو ادامه پیدا کند. همان گونه که در نمودار می‌بینید درآمد فروش دارو به صفر نمی‌رسد. چند دلیل برای این امر وجود دارد:

- آگاهی افراد، داروخانه‌ها و پزشکان از نام تجاری (اصلی) دارو
- اعتماد به شرکت اصلی عرضه کننده دارو
- درک کردن کیفیت واقعی و تأثیر بهتر محصول اصلی
- تأثیرات سود داروهای ژنریک بر برخی افراد
- این اعتقاد در برخی افراد که باید حق الزحمه شرکتهای دارویی تا زمان استفاده از دارو پرداخت شود

این است دلیل اینکه شرکتهای داروسازی محصول خود را گران‌تر از نسخه‌های ژنریک به خوبی به فروش می‌رسانند، اما در برخی از شرایط ممکن است شرکت تصمیم بگیرد قیمت محصول خود را به قیمت داروی ژنریک نزدیک‌تر کند.

راه دیگر حفظ درآمد یک محصول این است که طرحهای تجاری جدیدتری طراحی شوند یا کاربردهای جدیدی برای محصول پیدا شود. در مورد برخی از داروها این بدین معناست که مثلاً بازار آنها به ناحیه‌های دیگر دنیا گسترش داده شود یا آنها را به طور خلاقانه‌ای بتوان ترکیب کرد تا بیماری‌های دیگری را نیز بتوانند درمان کنند. بخاطر آورید بحث اول فصل را در زمینهٔ رانیتیدین و نام تجاری زانتاک. علت اینکه شما احتمالاً نام زانتاک را می‌شناسید این است که تولید کننده دارو، «گلاکسو اسمیت کلاین» نوع با دوز کمتری از این دارو (Zantac 75) را برای فروش بدون نسخه، عرضه کرد. به کمک تبلیغات گسترده فروش زانتاک بسیار موفق بود و امروزه میلیونها نفر آن را به عنوان بهترین و مؤثرترین آنتی اسید می‌شناسند. گرچه هم اکنون انواع ژنریک رانیتیدین در بازار موجود است، اما قدرت نام تجاری و انواع بدون نسخه این دارو همچنان درآمد عاید تولیدکننده اصلی آن می‌کند.

این است مدل تجاری خلاقانه‌ای که صنعت نرم‌افزار باید طراحی کند تا با آمدن محصولات متن‌باز متضرر نشود.

تأثیر متن‌باز بر نرم‌افزار

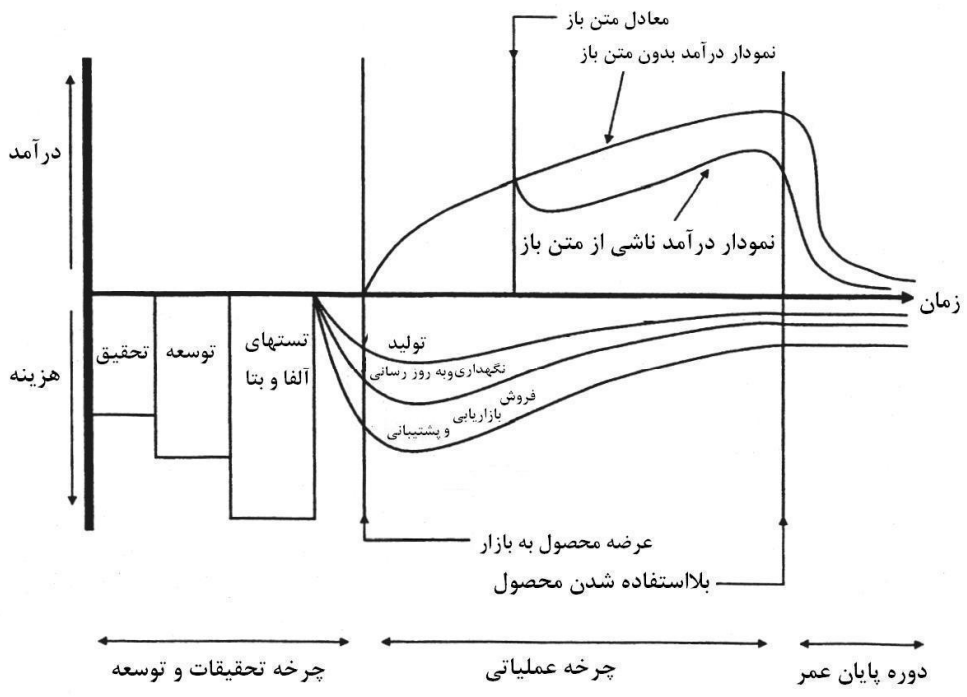
درد معدهٔ من چه ربطی به نرم‌افزار و متن‌باز دارد؟ گرچه تفاوتهای عمده‌ای بین این دو وجود دارد، اما تأثیر متن‌باز بر صنعت نرم‌افزارهای تجاری بسیار شبیه تأثیر داروهای ژنریک بر صنعت داروسازی است. صنعت داروسازی سالهاست که با این پدیده دست و پنجه نرم می‌کند و یاد گرفته است که چگونه مدلهای تجاری خود را متناسب با آن طرح‌ریزی کند. اما در صنعت نرم‌افزار، متن‌باز پدیدهٔ جدیدی است و ما هنوز در حال وفق دادن خود با تأثیرات آن هستیم.

در ابتدا نگاهی می‌اندازیم به تفاوتهای عمدهٔ بین این دو صنعت. یک محصول نرم‌افزاری معمولاً دارای مجموعه‌ای از امکانات و قابلیتها می‌باشد. در حالی که، یک دارو اغلب یک ترکیب شیمیایی مشخص است. ثبت اختراعی که از اجزای مشخصی تشکیل شده است، بسیار راحت‌تر از یک محصول نرم‌افزاری بزرگ است. عدهٔ زیادی در جامعهٔ متن‌باز که مخالف ذکر حق ثبت نرم‌افزار می‌باشد. این مخالفتها از آنجا ریشه می‌گیرد که در جامعهٔ متن‌باز مالکیت‌های معنوی آزادانه در اختیار سایر افراد قرار داده می‌شود و آنها همین را از بقیه انتظار دارند.

یک داروی ژنریک سعی می‌کند که مشابه محصول اصلی باشد، اما در رقابت بین محصولات نرم‌افزاری حتی اگر بین محصولات متن‌باز باشد، روشهای کاملاً متفاوتی وجود دارد. تفاوت اصلی در اینجاست که شرکتهای دارویی می‌توانند به مدت چند سال روی انحصاری بودن محصول خود حساب کنند، اما شرکتهای نرم‌افزاری به هیچ وجه نمی‌توانند انحصار دسته‌ای از محصولات را در دست گیرند.

قابلیت پیش‌بینی نیز در این دو مدل بسیار متفاوت است. صنعت داروسازی می‌تواند مدل هزینه‌ای بسیار دقیقی بر اساس عمر محصول خود بنا کند، اما در صنعت نرم‌افزار معلوم نیست چه زمان رقیبی محصولی بهتر یا ارزان‌تری عرضه خواهد کرد. اما صنعت داروسازی با همان فشارهای

رقابتی سر و کار دارد که صنعت نرم‌افزار با آنها دست و پنجه نرم می‌کند و به همین دلیل منحنیهای درآمد دقیق نیستند. شکل ۱۰-۲ همان مدل قبلی است که این بار مدلهای اقتصادی صنعت نرم‌افزار به آن اعمال شده است.



شکل ۱۰-۲ تاثیر متن باز بر ارزش محصول

جامعهٔ متن‌باز تلاش می‌کند نرم‌افزاری تولید کند که بیشترین خواهان را داشته باشد. این بدان معناست که اگر بازار محصول شما کوچک باشد، تعداد برنامه‌نویسان متن‌بازی که قصد کپی‌سازی نرم‌افزار شما را داشته باشد، بسیار کم است. اولین محصولات متن‌بازی که توسعه یافتند، کارگزارهای وب، موتورهای پایگاه اطلاعات، رابطهای رومیزی و ابزارهای برنامه‌نویسی بودند. تعدادی از محصولات و برنامه‌های کاربردی خاص نیز وجود دارد که هیچ معادل متن‌بازی ندارد. نکته در اینجاست که هر چه کاربرد محصول نرم‌افزاری گسترده‌تر باشد، احتمال اینکه معادل آن به صورت متن‌باز تهیه شود، بسیار بیشتر است.

البته در باور اینکه صِرفِ داشتن بازار کوچک، در معرض آسیب‌های متن‌باز نیستید، بسیار احتیاط کنید. به عنوان یک مثال خوب می‌توان به ابزار پردازش و روتوش عکس و فیلم اشاره کرد. جامعهٔ متن‌باز ابزار بسیار پیشرفته‌ای برای پردازش عکس‌ها تهیه کرده است، بنام GIMP و این

محصول در حال گسترش امکانات برای تدوین فیلم می‌باشد. گرچه GIMP ممکن است نیازهای همهٔ کاربران را برآورده نکند، اما معیار خوبیست از اینکه ارزش محصول شما به سرعت بخاطر فشارها کاهش می‌یابد.

شکل ۱۰-۲ تأثیرات متن‌باز را بر ارزش نرم‌افزار و قدرت قیمت‌گذاری شرکتها را نشان می‌دهد. تقریباً هر نرم‌افزار تجاری دارای ارزشی خاص خود است. از دید مشتری این طبیعی‌ست که ارزش نرم‌افزار با گذر زمان کاهش یابد. اما این کاهش ارزش نرم‌افزار، از فشارهای رقابتی و تا حدی تأثیرات خود محصول ناشی می‌شود. ۲۰ سال قبل شیشه‌های برقی برای اتومبیل گزینهٔ بسیار گران‌قیمتی بود. اما امروزه اگر این امکان در اتومبیلی وجود نداشته باشد، تعجب می‌کنید. فروشندگان نرم‌افزار (و سازندگان اتومبیل) عموماً به این فرآیند طبیعی کاهش ارزش این گونه واکنش نشان می‌دهند که به طور پیوسته امکانات و قابلیتهای جدیدی به محصول خود اضافه می‌کنند و محصول خود را به روز می‌کنند. این به‌روزرسانی محصول، ارزش آن را تجدید می‌کند. در زمینه نرم‌افزارهای شرکتی، فروشندگان این گونه نرم‌افزارها به این ترتیب می‌توانند ارزش آن را حفظ کنند که تنها مرجع قادر به پشتیبانی آن نرم‌افزار باشند. راه دیگر این است که راه ورود رقبا را ببندند. وقتی که یک مشتری نرم‌افزار شما را انتخاب می‌کند، مادامی که هزینه‌های پشتیبانی و بروزرسانی آن کمتر از هزینه جایگزینی آن باشد، درآمد پیوسته‌ای عاید شما می‌شود و راه ورود رقبا بسته می‌شود.

متن‌باز از چندین جهت فرآیند کاهش ارزش را شتاب می‌دهد. اولاً، از آنجایی که ممکن است چندین تولیدکننده یک مشکل را هدف قرار دهند، شما زمان بسیار کمی برای تولید یک محصول قابل رقابت دارید. ثانیاً، از آنجایی که جامعهٔ متن‌باز با پول تحریک نمی‌شود، همه خدماتی که شما با دریافت هزینه‌ای در اختیار قرار دهید، آنها به طور رایگان انجام می‌دهند. ثالثاً، حتی اگر به روش ذکر شده، مانع برای ورود رقبا ایجاد کنید، این مانع در مقابل متن‌باز بی‌تأثیر است. زیرا هزینهٔ خرید این نرم‌افزارها اغلب صفر است. بنابراین این مانع فقط می‌تواند از جنبهٔ پشتیبانی محصول، در مقابل

متن‌باز بایستد. به هر حال شدت و ضعف این مانع بستگی به میزان علاقهٔ کاربر به روشهای پشتیبانی متن‌باز دارد.

مشکل آخر هم قابلیت پیش‌بینی است. صنعت داروسازی می‌تواند با دقت بالا، زمان عمومی شدن محصول را پیش‌بینی کند و با مدلهای هزینه‌ای دقیقی می‌تواند در مقابل آن واکنش نشان دهد. اما تأثیرات متن‌باز بسیار غیرقابل پیش‌بینی است؛ اولاً اگر فعالیتهای جاری جامعهٔ متن‌باز را زیر نظر نداشته باشید، ممکن است هنگام عرضه محصول خود متوجه شوید قبلاً نسخهٔ متن‌باز آن آمده است. ثانیاً از آنجایی که محصولات نرم‌افزاری معمولاً ملغمه‌ای از امکانات و ویژگیها هستند، نمی‌توان تعیین کرد که چه زمانی یک محصول متن‌باز موجود، از دید مشتریان، مقبول خواهد بود.

کاهش ارزش به عنوان یک مزیت رقابتی

کلید بهره بردن از متن‌باز این است که یاد بگیریم چگونه از کاهش ارزش محصولات و امکانات جامعهٔ متن‌باز حداکثر استفاده را ببریم، به جای اینکه حالت تدافعی نسبت به آنها اتخاذ کنیم. در اینجا تعدادی از مثالهای واکنشهای دفاعی نسبت به متن‌باز ذکر شده است:

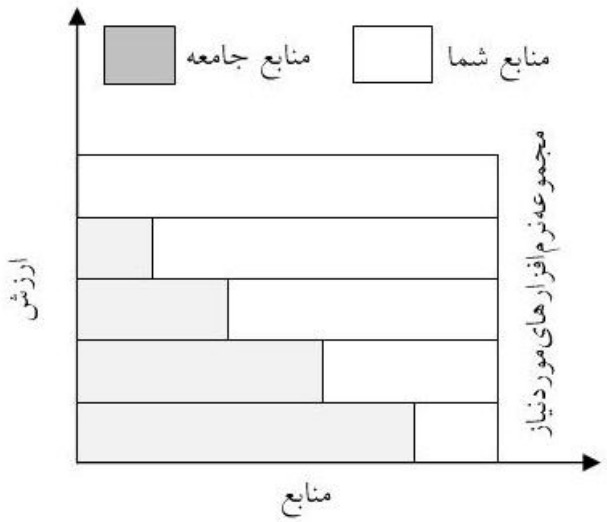
- **استفاده سخت‌گیرانه از حق ثبت**- برخی از شرکتهای نرم‌افزاری با سلاح حق ثبت به جامعهٔ متن‌باز حمله می‌کنند. این تاکتیک توصیه نمی‌شود. از آنجایی که جامعهٔ متن‌باز از منابع بسیار ضعیف و محدودی برای دفاع از خود در دادگاه برخوردار است، اغلب از طریق کشاندن مسأله به رسانه‌ها و تخریب شاکی از خود دفاع می‌کند. تقریباً در اکثر موارد این امکان وجود دارد که کد برنامه چنان نوشته شود که از نظر قانونی به حق ثبت برنامه دیگری تعدّی نکند و بنابراین در نهایت چیزی عاید شاکی نخواهد شد. شرکت Unisys انحصار الگوریتم فشرده‌سازی تصویر را که برای فایلهای GIF استفاده می‌شود، به نام خود ثبت کرد. در آن زمان، از این قالب گرافیکی به طور گسترده‌ای در اینترنت استفاده می‌شد. جامعه متن باز این گونه به این اقدام پاسخ داد که از قالبهای JPEG و PNG به جای GIF استفاده کرد.

- **تهیه نرم‌افزار به صورت یکپارچه-** امروزه اکثر نرم‌افزارها به صورت لایه‌های عملیاتی نوشته می‌شوند. هر لایه بر روی لایه دیگری بنا می‌شود. برنامه‌نویسان نرم‌افزارهای متن‌باز، نوعاً کدنویسی برنامهٔ خود را از پائین‌ترین لایه این پُشته آغاز می‌کنند. برخی از تولیدکنندگان نرم‌افزار سعی می‌کنند با نوشتن برنامه به صورت یکپارچه و یکجا جلوی عمومی شدن آن را بگیرند. اما نتیجه این تلاش این می‌شود که جامعه متن‌باز نرم‌افزار مشابهی با معماری بهتر طراحی می‌کند و تمام امکانات نرم‌افزار یکپارچه قبلی را در اختیار عموم قرار می‌دهند.
- **رقابت-** یکی دیگر از واکنشها، تلاش برای رقابت با جامعهٔ متن‌باز است. اما اثرات این واکنش کاملاً با گذر زمان کمرنگ می‌شود. رقابت با جامعه‌ای از برنامه‌نویسان خبره که انگیزه‌شان پول نیست و از نظر زمانی محدود نیستند، اشتباه بزرگی است. این رقابت مستلزم صرف هزینه بسیار سنگین است و معمولاً زودتر از آنچه که تصورش را بکنید از ادامه رقابت منصرف خواهید شد.

بهترین گزینه این است که به طور فعال به ایفای نقش در این جامعه بپردازید و از کم ارزش شدن محصولات به نفع خود بهره‌برداری کنید. در فصل بعد، برخی از مدلهای تجاری دیگر را بررسی خواهیم کرد. در اینجا فقط بر زیرساختهای فرهنگی و فکری که برای کار با این جامعه مورد نیاز است، متمرکز خواهیم شد. این فرهنگ چنان باید باشد که با دید مثبتی به جامعهٔ متن‌باز و تحولات آن نگریسته شود.

بدون تأثیرات جامعه متن‌باز نیز محصول شما، به مرور زمان دستخوش کاهش ارزش خواهد شد. بنابراین شما باید همواره فناوری را چنان گسترش دهید که در قلهٔ ارزش نرم‌افزارها قرار داشته باشید و در این راستا مصرف منابع بالایی نیز خواهید داشت. اگر منابع خود را صرف فناوری کنید که نزد مشتریان از ارزش پایینی برخوردار است، سرمایه شما برنمی‌گردد.

همهٔ شرکت‌های نرم‌افزاری از منابع محدودی برخوردارند. در اکثر موارد شرکت‌ها منابع‌شان را صرف بالا رفتن از قلهٔ ارزش نرم‌افزارها می‌کنند. اما ترفندی که باید در اینجا به کار گرفته شود این است که با جامعهٔ متن‌باز شریک شوید تا بتوانید از منابع آن بهره گیرید. سپس از آن منابع برای تولید محصول باارزش‌تر استفاده کنید. در شکل ۱۰-۳ نسبت استفاده از منابع جامعهٔ متن‌باز و شرکت برای عرضهٔ یک محصول دارای ارزش خاص، نمایش داده شده است.



شکل ۱۰-۳: نمودار رابطه منابع با ارزش آنها

سیاست شما باید این باشد که با بالا رفتن ارزش محصول، منابع بیشتری به آن اختصاص دهید. در فصل بعدی در زمینهٔ نحوهٔ ترکیب نرم‌افزارهای اختصاصی و متن‌باز بحث خواهد شد. نکته‌ای که باید به آن دقت شود این است که حتی در پایین‌ترین سطح ارزش نرم‌افزار، باید قسمتی از منابع خود را به آن اختصاص دهیم. نباید چنین تصور کنید که می‌توان نرم‌افزار را به جامعه سپرد و در عین حال بر آن نظارت داشت. باید حداقل یک نفر از شرکت شما پروژه را هدایت و فعالیت جامعهٔ متن‌باز را با اهداف شرکت همراستا کند. او همچنین باید مطمئن شود که لایهٔ نرم‌افزار در حال توسعه، نیازهای لایهٔ بالاتر را برآورده می‌کند. در نهایت باید خودتان نگهدارنده آن نرم‌افزار باشید.

مزیت دیگر حفظ ارتباط نزدیک با جامعهٔ متن‌باز این است که در تمامی سطوح توسعهٔ نرم‌افزار از نحوهٔ تأثیر احتمالی جامعه بر ارزش نرم‌افزار خود آگاه می‌شوید و می‌توانید به موقع جهت‌گیری پروژه را تغییر دهید. بنابراین معیارهای دقیقی از نحوهٔ پیشبرد پروژه در دست خواهید داشت.

ارزش در حصار زمان

در یکی از کنفرانس‌های Linux World، من با «تیم اورایلی» (مدیر انتشارات «O'Reilly») برای صرف نوشیدنی همراه شدم. ما دربارهٔ مسائل مختلف متن‌باز صحبت کردیم و من عبارت «ارزش به عنوان تابعی از زمان» را به کار بردم و مفهوم این عبارت را برای تیم توضیح دادم. سپس او دیدگاه خودش را از مفهوم این عبارت برای من توضیح داد که با دیدگاه من متفاوت بود. نظر او این بود که جامعهٔ متن‌باز برای عرضه یک محصول مفید و کارا، محدود به یک بازه زمانی خاص نیست. هر کس که با صنعت فناوری‌های پیشرفته در ارتباط باشد، به خوبی می‌داند که محصولات باید قبل از موعدشان عرضه شود. اگر این زمان بگذرد، این محصول دیگر روشنایی روز را نخواهد دید و در انبارها خاک خواهد خورد. جامعهٔ نرم‌افزارهای آزاد درگیر چنین محدودیتهایی نیست و هر فناوری در هر زمانی استفادهٔ خودش را پیدا خواهد کرد. پس از این ملاقات، توضیحات او فکر مرا به خود مشغول کرد و به این نتیجه رسیدم که خود من هم درگیر مسألهٔ «ارزش در حصار زمان» بوده‌ام. در ادامه تجربه خود را در این زمینه بازگو خواهم کرد.

کلاسهای بنیادی Pilot

در سال ۱۹۹۶، شرکت US Robotics یک PDA[1] (دستگاه ذخیره اطلاعات دستی) استثنائی به نام Pilot 5000 روانه بازار کرد. امروزه ما تکامل یافته همان محصول را با نام Palm می‌شناسیم. زمانی که این محصول عرضه شد من بلافاصله یکی از آن را تهیه کردم و تصمیم گرفتم آن را برنامه‌ریزی کنم. سیستم‌عامل Pilot (که امروزه Palm OS نامیده می‌شود) بر اساس مجموعه

[1] . Personal Digital Assistant

توابعی به زبان C بنا شده است. من با برنامه‌نویسی شیءگرا آشنا بودم و قصد داشتم این دستگاه را با زبان C++ برنامه‌ریزی کنم. این کار از نظر فنی امکانپذیر بود، اما وقتی عملی می‌شد که من می‌توانستم از توابع Palm OS به عنوان مجموعه‌ای از کلاس‌های C++ در برنامه‌های خود استفاده کنم. من باید تقریباً همان کاری را می‌کردم که میکروسافت برای ساختن کلاس‌های بنیادی خود (MFC) انجام داده بود. MFC مجموعه‌ای از کتابخانه‌هاست که برنامه‌نویسی ویندوز را بسیار آسانتر می‌کند. به این ترتیب که توابع Win32 را به صورت کلاس‌هایی در اختیار برنامه‌نویسان قرار می‌دهد.

در نهایت خودم این مجموعه کلاس‌ها را برای Palm OS نوشتم و آن را کلاسهای بنیادی Pilot نامیدم. من این پروژه را به صورت پاره‌وقت، در حین سفرهایم به آن سوی اقیانوس اطلس، انجام می‌دادم. گرچه من این پروژه را تکمیل کرده بودم اما برنامه‌نویس دیگری نیز پروژه مشابهی را تکمیل و عرضه کرد. من دلیلی نمی‌بینم که بخواهم با او رقابت کنم. من به اندازه کافی بستر Palm OS را شناخته بودم و از وقتی که صرف آن کرده بودم، راضی بودم. حداقل چیزی که برای من ماند تجربه‌ای بود که از نوشتن کلاس‌های بنیادی، بدست آوردم.

Jump

یکی از مهم‌ترین کسانی‌که به محیط Palm خدمت کرد، گِرِگ هیوجیل[1] بود. گِرِگ یک شبیه‌ساز بسیار جالب با نام Co Pilot عرضه کرده بود که نقطهٔ اتکاء هر برنامه‌نویسی بود (در فصل بعد این شبیه‌ساز را بررسی خواهیم کرد). حدود سال ۱۹۹۷ گِرِگ پروژه دیگری را به نام Jump آغاز کرد. اگرچه در تعریف رسمی Jump گفته شده است که از ابتدای کلمات Java User Module for Pilot گرفته شده است، اما گِرِگ ادعا می‌کند که این نام را از آهنگی که موقع هَک کردن گوش می‌داد، گرفته است. Jump به برنامه‌نویسان این امکان را می‌داد که برنامه‌های خود را به زبان جاوا بنویسند و سپس به کمک آن برنامهٔ خود را به کد ماشین دستگاه‌های Palm تبدیل کنند. آنها که

[1] . Greg Hewgill

با زبان جاوا آشنایی دارند می‌دانند که برنامه‌های جاوا اغلب توسط یک ماشین مجازی به نام JVM (ماشین مجازی جاوا) اجرا می‌شوند. اما Jump برنامه‌های جاوا را مستقیماً تبدیل به کد ماشین پردازندهٔ موتورولا 68k می‌کند. وقتی که گِرِگ در حال آماده کردن نسخهٔ بتای Jump بود، من روی برنامهٔ نمایشی Word Clock کار می‌کردم که بعداً نام آن را به Desk Clock تغییر دادم (در شکل ۱۰-۴ نمونه‌ای از خروجی این برنامه‌ها را می‌بینید). این برنامه ساعت سه منطقه زمانی را به طور همزمان بر روی صفحهٔ نمایش نشان می‌داد. گِرِگ قصد داشت برنامه‌های نمایشی مرا به عنوان برنامه‌های کمک آموزشی به همراه Jump ارائه کند.

شکل ۱۰-۴: نمونه ای شبیه ساز PalmOS درحال اجرای DeskClock

اینجا بود که من به مهارت کسب شده در طراحی کلاس‌های بنیادی مراجعه کردم. Java زبانی شکل‌گراست و طراحی کلاس‌های بنیادی Java برای توابع Palm OS هدف بعدی من بود. این بار ما چند نفر بودیم که بر روی PilotJFC کار می‌کردیم. اما این‌بار بیشتر از منابع جامع متن‌باز استفاده کردیم. این است نمونه‌ای از فناوری‌ای که من طراحی کردم و چنان مورد استفاده برنامه‌نویسان قرار گرفت که فکرش را هم نمی‌کردم.

گرچه به دلایل نامشخصی، گِرِگ کار بر روی Jump را به طور ناگهانی رها کرد. Jump محدودیتهایی داشت که استفاده از آن را به عنوان یک بستر برنامه‌نویسی مشکل می‌ساخت. به

تدریج جامعه‌ای که روی Jump کار می‌کرد از هم پاشید. گِرِگ متقاعد شد از آنجایی که دیگر روی Jump کار نمی‌کند، باید کد منبع آن را منتشر کند تا بقیه و افراد دیگری کار او را ادامه دهند. او Jump را تحت مجوز GPL عرضه کرد. من هم دیگر سیر تحولات Jump را پیگیری نکردم. چند سال قبل برنامه‌نویس دیگری به نام «رالف کلبرف» پروژه Jump را با نام Jump2 تکمیل کرد.

تصور کنید من چقدر هیجان‌زده شدم وقتی که بعد از ۴ سال، ارتباط نداشتن با برنامه‌نویسی، ایمیل زیر را از جیم برچفیلد دریافت کردم. او این ایمیل را برای برنامه‌نویس JFC دیگری نیز فرستاده بود.

از: جیم برچفیلد

تاریخ ارسال: سه شنبه ۲۷ فوریه ۲۰۰۱، ساعت: ۱:۳۵ بعدازظهر

به: مارتین فینک

موضوع: احیاء Pilot JFC

من اخیراً شروع به کار با Jump کرده‌ام و متوجه شدم که در برخی از اظهارنظرهای قدیمی، به Pilot JFC اشاره شده است.

من نسخهٔ قبلی JFC را ارتقاء داده‌ام و آن را با آخرین نسخهٔ Jump منطبق کرده‌ام. من Word Clock را نیز به این معماری جدید منتقل کرده‌ام. در حین این کار برخی از امکانات آن را بهبود داده‌ام. البته باید بگویم که آخرین نسخهٔ آن نیز به خوبی کار می‌کند.

آیا هیچ یک از شما می‌توانید مرا در جریان وضعیت کنونی پروژه Pilot JFC قرار دهید؟ من قصد دارم این پروژه را به صورت امکانات اضافهٔ توزیع Jump عرضه کنم. همچنین مایلم توسعه و نگهداری آن را نیز بر عهده گیرم. اما دوست ندارم ببینم نسخه‌های مختلف و ناهمگنی از آن عرضه شده است.

بنابراین من می‌خواهم کد منبع آن را در اختیار داشته باشم، با این شرط که همه جا نام شما را ذکر کنم. همچنین تعدادی برنامهٔ نمایشی خواهم نوشت و آنها را از اجزاء اصلی Pilot JFC قرار خواهم داد.

تیم اورایلی باعث شد که من متوجه موضوعی شوم که خودم قبلاً آن را تجربه کرده بودم. نرم‌افزاری که توسط چندین نفر نوشته شده بود، آنقدر خاک خورد تا اینکه نیاز به آن از طرف جامعهٔ کاربران احساس شد.

این مثال باید برای همکاری بیشتر با جامعه متن‌باز انگیزهٔ بیشتری در شما ایجاد کند. حتی اگر پروژه‌های برنامه‌نویسی شما متوقف شده‌اند یا با سرعت بسیار کمی پیش می‌روند، آنها را دست کم نگیرید. تنها چیزی که آنها احتیاج دارند یک برنامه‌نویس با انگیزه است تا فعالیت روی آن را از سر گیرد و آن را به عنوان یک محصول متن‌باز در اختیار عموم قرار دهد. شاید روزی خود شما تبدیل به آن فرد با انگیزه شوید و اجرای پروژه را سرعت بخشید. به عنوان حسن ختام به چند نکته در زمینه ایمیل جیم اشاره می‌کنم. اولاً او قبل از بدست گرفتن کاری از نویسندگان اصلی آن اجازه گرفت. ثانیاً او به من اطمینان داد که نام مرا در کارهایش ذکر می‌کند. ثالثاً دقت کنید که او از کلمهٔ توزیع برای Jump به همان مفهومی که من در توزیعهای غیرلینوکسی در فصل ۵ توضیح دادم، استفاده کرده است. نهایتاً اینکه او قصد ندارد که با سایر تلاش‌های انجام شده در جامعه رقابت کند. همین ایمیل کوتاه توانسته است نمایانگر فرهنگ جامعهٔ متن‌باز باشد.

جمع‌بندی

در این فصل تاثیرات اقتصادی نرم‌افزارهای متن‌باز بر محصولات شما شرح داده شد. در پاسخ به تاثیرات آنها شما می‌توانید به آنها حمله کنید، دفاع کنید یا حتی با آنها رقابت کنید. اما بهترین را این است که با این جامعه همکاری و شراکت کنید و با استفاده از شرایط به وجود آمده، منابع خود را بر فعالیت‌هایی متمرکز کنید که منفعت بیشتری برای شما داشته باشد. مدیران فناوری اطلاعات نیز با استفاده از شرایط بوجود آمده باید برسرعت پروژه‌های خود بیفزایند. همچنین دیدید

که همکاری کردن با جامعه باعث می‌شود با پروژه‌های در حال انجام و حتی متوقف شده آشنا شوید. بدین ترتیب می‌توانید مسیر تکامل مدل تجاری شرکت خود را بهتر ترسیم کنید.

حال که آثار اقتصادی متن‌باز و بهترین پاسخ نسبت به آن را یاد گرفتیم، زمان آن رسیده است که نحوهٔ تکامل مدل تجاری شرکت را بررسی کنیم. در فصل بعد جزئیات مدلهای تجاری که می‌توانند از متن‌باز بیشترین بهره را ببرند، بررسی خواهد شد.

فصل ۱۱

مدلهای تجاری درآمدزا

به عنوان یک تولیدکننده محصولات نرم‌افزاری، بی‌شک زمانی فرا خواهد رسید که متوجه خواهید شد لینوکس و متن‌باز تأثیر مستقیمی بر جریان درآمد شما می‌گذارد. در این شرایط باید نحوهٔ توسعهٔ محصولات را اصلاح کنید و به منابع درآمدی خود از جنبهٔ دیگری نگاه کنید. همچنان که با لینوکس، متن‌باز و مفهوم نرم‌افزار آزاد آشنا می‌شوید، باید نحوهٔ درآمدزایی را با تولید محصولات، خدمات و پشتیبانی‌های جدید یاد بگیرید. در این فصل چند مدل تجاری بررسی خواهد شد. از آنجایی که کسب و کارها متفاوت هستند، هر یک از این مدلها را باید با شرایط کسب و کار خود سازگار کنید. هدف این فصل این است که چندین مدل تجاری و نحوهٔ استفاده از آنها در کسب و کار خود را به شما بیاموزد.

این فصل نیز برای شرکت‌های تولیدکننده محصولات نرم‌افزاری نوشته شده است، اما همچنان برای مدیران فناوری اطلاعات مفید است. تحول در مدل تجاری تولیدکنندگان نرم‌افزار بدون شک بر نحوهٔ خرید مدیران فناوری اطلاعات نیز تأثیر می‌گذارد. مدلهای تجاری که در یک فصل بررسی خواهند شد عبارتند از:

- نرم‌افزار تجاری برای لینوکس
- خدمات و پشتیبانی یک محصول متن‌باز
- ارتقاء و بهبود محصولات تجاری به کمک نرم‌افزارهای متن‌باز
- نرم‌افزار متن‌باز برای فعال‌سازی اجزای سخت‌افزاری
- تجاری کردن یک محصول متن‌باز از طریق مجوز دوگانه
- کاهش هزینه پایان چرخهٔ زندگی
- ایجاد یک محیط

جنبش نرم‌افزار آزاد و اکثر فعالان جامعهٔ متن‌باز معتقدند که نرم‌افزار باید آزاد باشد. گرچه منظور از آزادی در دسترس بودن کد منبع است، اما در اغلب موارد به مفهوم رایگان نیز می‌باشد. در این فصل قصد نداریم بگوییم نرم‌افزارهای شما باید رایگان باشند. متن‌باز در صورتی که نحوهٔ پیاده‌سازی آن را بلد باشید، می‌تواند در فرآیند بسیار مفید واقع شود.

این فصل جزئیات مدلهای تجاری مختلف را بررسی خواهد کرد. در فصل بعد فرآیند استفاده از متن‌باز در محصولات داخلی و عرضهٔ یک نرم‌افزار متن‌باز را به جامعه بررسی خواهیم کرد.

ارزش محصول خود را بدانید

اولین قدمی که باید بردارید این است که ماهیت کسب و کار خود را روشن کنید. مشتریان شما چه کسانی هستند و چه چیز با ارزشی را به آنها عرضه می‌کنید؟ در فصل گذشته یاد گرفتید که هر محصول با ارزشی با گذر زمان عمومی خواهد شد و از انحصار شما درخواهد آمد. متن‌باز به فرآیند عمومی شدن شتاب بیشتری می‌دهد. بنابراین هدف شما باید این باشد که همواره محصول باارزش‌تری به مشتریان خود عرضه کنید. این بدین معناست که اگر محصول شما در حد بهترین محصول موجود نباشد، مشتریان آن را نمی‌خرند.

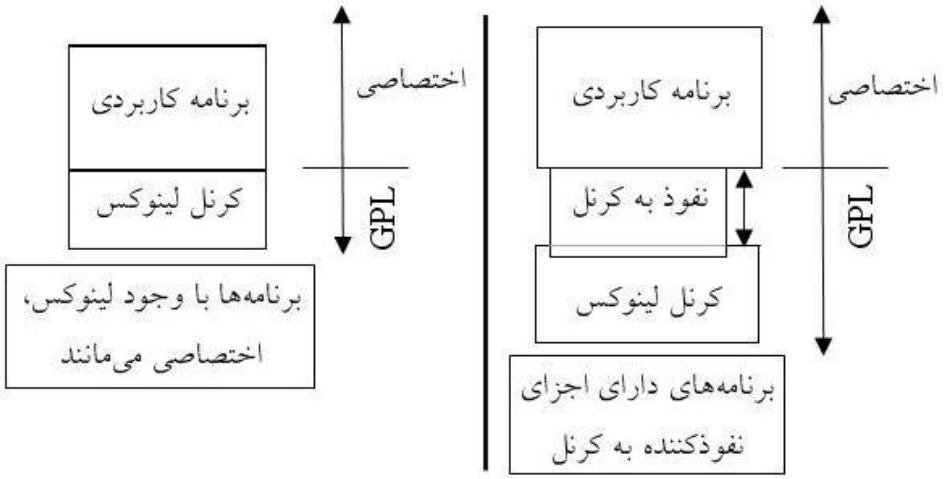
اگر از جایگاه خود نزد مشتریان آگاهی داشته باشید به راحتی می‌توانید آنچه را که مورد نیاز کسب و کارتان است، تشخیص دهید و بخشهایی را که دیگر نمی‌تواند برای شما پول‌ساز باشند عمومی کنید. متن‌باز از این طریق می‌تواند تأثیر مثبتی بر کسب و کار شما داشته باشد. با بررسی و ارزیابی دقیق جامعهٔ متن‌باز، شما می‌توانید همواره روی باارزش‌ترین محصول سرمایه‌گذاری کنید. این فرآیند می‌تواند مدیران شما را دستپاچه کند. آنها از این پس باید منابع تولید محصول خود را از جاهای دیگری تأمین کنند. اصلاح فرهنگی که باید در سازمان صورت دهید، این است که مدیران و کارمندانشان به طور پیوسته در فکر تولید محصولات باارزش‌تر باشند و در عین حال همکاری با جامعهٔ متن‌باز نیز تشویق شود.

با فرض اینکه شما سیاستهای تجاری سازمان و اصل ارتقاء ارزش محصول را به درستی درک کرده‌اید، چند مدل تجاری برای پیاده کردن در سازمان در اختیار خواهید داشت.

نرم‌افزار تجاری و لینوکس

این ساده‌ترین مدل تجاری مطرح شده در این فصل است. با این حال باید به چند نکته اشاره شود. بسیاری افراد تصور می‌کنند که نمی‌توان نرم‌افزارهای تجاری و اختصاصی را تحت لینوکس اجرا کرد. این تصور غلط است. به غیر از چند استثناء، نرم‌افزارهای کاربردی اختصاصی تحت لینوکس را همانند نرم‌افزارهای اختصاصی سایر سیستم‌عاملها، می‌توان به فروش رساند. امروزه به نمونه‌های زیادی از نرم‌افزارهای تجاری تحت لینوکس می‌توان اشاره کرد. در مجوز لینوکس (GPL) هیچ الزامی برای انتشار کد منبع این گونه نرم‌افزارها نشده است. در شکل ۱۱-۱ شمائی از نحوهٔ اجرای نرم‌افزارهای اختصاصی و نرم‌افزارهای نفوذکننده به کِرنل در لینوکس دیده می‌شود.

برخی از نرم‌افزارهای تجاری از مرزهای کِرنل لینوکس فراتر می‌روند. در این گونه نرم‌افزارهای خاص باید فرآیند تولید و یکپارچه‌سازی نرم‌افزار به صورت دیگری انجام شود. در اینجا باید چند تصمیم‌گیری تجاری انجام دهید. از آنجایی که هستهٔ لینوکس به صورت GPL مجوزدهی شده است، قسمتهایی از نرم‌افزار شما که در کِرنل انجام می‌شود باید تحت GPL مجوزدهی شود. توصیه می‌کنم فصلهای ۲ و ۳ را، برای تصمیم‌گیری در زمینه ماجول‌های کرنل و مجوز GPL مرور کنید.



شکل ۱۱-۱: اجرای نرم‌افزارهای تجاری در لینوکس

اگر بدانید که از دید مشتری ارزش محصول شما کجاست، متوجه می‌شوید که اجزاء نفوذکننده در کِرنل برای مشتریان شما ارزشی ندارند. بنابراین این اجزاء را می‌توانید به صورت GPL عرضه کنید. اما اگر کسب و کار شما به هر دلیلی اجازه ندهد که هیچ قسمتی از نرم‌افزار نفوذکننده به کرنِل GPL شود، نمی‌توانید آن را برای لینوکس عرضه کنید. در فصل ۲ آموختید که می‌توان ماجول‌های اختصاصی برای لینوکس تهیه کرد. اما این روش توصیه نمی‌شود، بنابراین آن را بررسی نمی‌کنیم. اگر مرز بین بخشهای نفوذکننده به کرنل و سایر بخشهای نرم‌افزار مشخص شده باشند، بدون هیچ مشکلی می‌توانید نرم‌افزار خود را عرضه کنید. بخشهای نفوذکننده به کِرنل، به صورت ماجول‌ها یا وصله‌هایی به کِرنل لینوکس عرضه می‌شود. سپس می‌توانید قسمتهای اختصاصی نرم‌افزار خود را به این ماجول‌ها متصل کنید و محصول را به مشتری خود عرضه کنید.

پشتیبانی و خدمات نرم‌افزارهای متن‌باز

توزیع‌کنندگان لینوکس یکی از اولین مدلهای تجاری مورد استفاده در جامعهٔ متن‌باز را طراحی کرده‌اند. ابتدایی‌ترین گونه این مدل تجاری این است که توزیع‌کنندگان لینوکس، پشتیبانی توزیع لینوکس خود را به مشتریان می‌فروشند. از آنجایی که توزیع‌کنندگان لینوکس توزیع خود را

با اجزاء متن‌باز دیگر که خودشان طراحی کرده‌اند عرضه می‌کنند، در زمینه یکپارچه‌سازی تبحر زیادی دارند. این توزیعهای متن‌باز چندین مدل تجاری را ایجاد کرده‌اند.

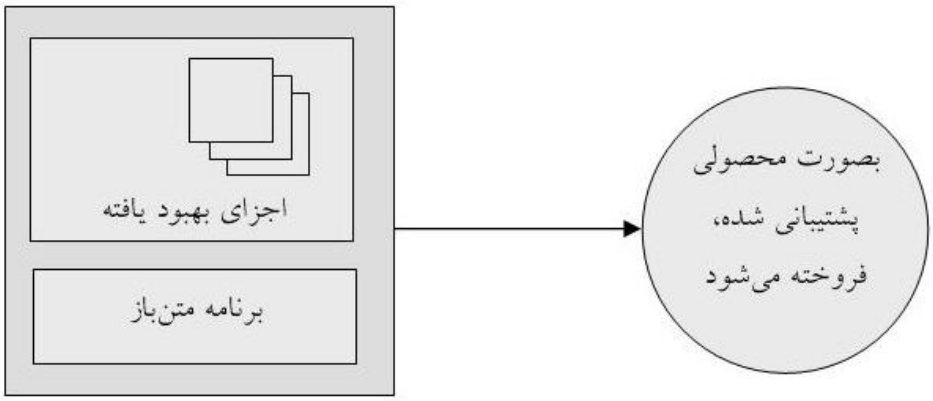
- **تهیه بسته‌های نرم‌افزاری برای فروش**- بسیاری از مشتریان ترجیح می‌دهند رسانه حاوی نرم‌افزار و راهنماهای آن را با پرداخت هزینه‌ای دریافت کنند. آنها نمی‌دانند که یک کپی از آن نرم‌افزار را می‌توان کاملاً قانونی و رایگان از طریق اینترنت دریافت کنند. بنابراین بسیاری از توزیع‌کنندگان لینوکس توزیع خود را به صورت بسته‌های نرم‌افزاری برای فروش عرضه می‌کنند.
- **پشتیبانی**- توزیع‌کنندگان لینوکس اغلب خدمات پشتیبانی خود را نیز به همراه بسته نرم‌افزاری عرضه می‌کنند تا بتوانند مشتری را در حل مشکلات پیش آمده کمک کنند. مشتریان شرکتی بزرگ خدمات پشتیبانی گسترده‌تری را برای لینوکس‌های خود درخواست می‌کنند. همچنین برخی از شرکت‌هایی که توزیع‌کننده لینوکس نیستند، مانند Linux Care، خدمات پشتیبانی برای طیف گسترده‌ای از محصولات متن‌باز ارائه می‌کنند.
- **آبونمان**- آبونمان بخشی از مجموعه خدماتی است که توسط برخی از توزیع‌کنندگان عرضه می‌شود. خدمات آبونمان امکاناتی را در اختیار مشتری قرار می‌دهد که همواره نرم‌افزار خود را به روز نگه دارند. شرکت زیمیان (Ximian) از خدمات آبونمان برای به روز کردن محیط رومیزی گِنوم استفاده می‌کند.
- **نام تجاری**- برخی از فروشندگان نرم‌افزار و سخت‌افزار از توزیع لینوکس خاصی استفاده می‌کنند و آن را با نام تجاری خود مرتبط می‌کنند و در فعالیت‌ها و تبلیغات نام آن را ذکر می‌کنند. بستگی به شرایط، فروشنده ممکن است برای این کار هزینه‌ای نیز دریافت کند. این مرتبط‌سازی باعث کسب اعتبار بیشتر برای آن توزیع یا فروشنده می‌شود و در جذب مشتریان مؤثر است.

- **خدمات حرفه‌ای-** کار پیوسته روی یک توزیع لینوکس، قدرت رقابتی توزیع‌کننده را بهتر نمایش می‌دهد. مثلاً، شرکت Red Hat در زمینه ارائه خدمات حرفه‌ای برای لینوکس خود تخصص دارد.

برای اینکه بتوانید مدلهای تجاری فوق را پیاده‌سازی کنید، ابتدا لازم است کسب و کار خود را به طور دقیق تحلیل کنید و به کمک آن هزینه‌های کسب اعتبار و درآمد ناشی از آن را تخمین زنید. برخی از شرکت‌ها به این نتیجه رسیده‌اند که ادامه دادن تجارت با این روش مشکل است و در حال مدل تجاری کنونی خود (مبتنی بر متن‌باز) با مدلهای سنتی (مثلاً فروش نرم‌افزارهای اختصاصی) هستند.

ترکیب و بالا بردن ارزش

در این مدل تجاری نرم‌افزارهای متن‌باز با نرم‌افزارهای اختصاصی به گونه‌ای ترکیب می‌شوند که راه حل‌های بهتری در اختیار مشتریان قرار دهند. در برخی موارد نرم‌افزارهای متن‌باز با یک محصول سخت‌افزاری ترکیب می‌شوند (در زمینهٔ مدل تجاری سخت‌افزار، در همین فصل توضیح خواهم داد). اما در اکثر موارد شما می‌توانید نرم‌افزار اختصاصی خود را به کمک اجزای متن‌باز بهبود دهید. راهکار دیگر این است که امکانات یک نرم‌افزار اختصاصی را به نرم‌افزاری متن‌باز اضافه کنید و بستهٔ نرم‌افزاری حاصل را بفروشید. شکل ۱۱-۲ نحوهٔ ترکیب یک نرم‌افزار متن‌باز با اجزاء اختصاصی را نشان می‌دهد.



شکل ۱۱-۲: بهبود دادن یک نرم‌افزار متن باز به کمک اجزای اختصاصی

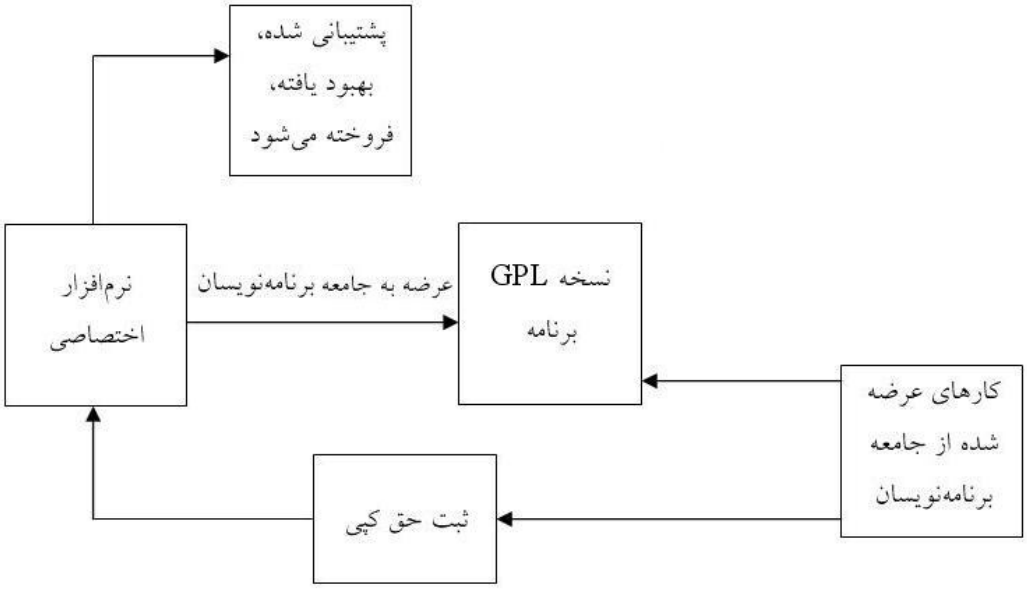
به عنوان یک مثال خوب از این مدل تجاری می‌توان به محصولی از شرکت فناوری «کووالنت» اشاره کرد. کارگزار وب آپاچی پر استفاده‌ترین کارگزار وب در اینترنت است و به نرم‌افزار متن‌باز کُشنده معروف است. کووالنت امکانات مورد نیاز شرکت‌های بزرگ را به این کارگزار می‌افزاید. این شرکت امکانات مدیریتی، امنیتی و غیره را به این کارگزار می‌افزاید و آن را آمادهٔ استفاده تجاری می‌کند. از محصولات دیگر کووالنت، خدمات پشتیبانی برای شرکت‌های بزرگ است. از آنجایی که تیم برنامه‌نویسان این شرکت، ارتباط نزدیکی با جامعهٔ متن‌باز و شرکت‌های تجاری دارند، پل ارتباطی مناسبی برای ارتباط مشتریان با این جامعه هستند.

برای هر شرکتی این امکان وجود دارد که امکانات تجاری به کارگزار آپاچی اضافه کند و حتی به انجام این کار با سایر شرکت‌ها رقابت کند. اگر این کار درست انجام شود، در حالی که شرکت‌ها برای عرضه کارایی با ارزش بالاتر رقابت می‌کنند، منجر به بهبود پیوسته این کارگزار نیز خواهد شد. این شرکت‌ها می‌دانند ارزشی که به آپاچی افزوده شده است، ناشی از ابزاریست که آنها طراحی کرده‌اند.

این مدل را می‌توان به هر یک از بسترهای اصلی متن‌باز اعمال کرد مانند کارگزارهای اینترنتی، فناوری‌های خوشه‌سازی (کلاسترینگ)، سیستمهای فایلی، رابط‌های کاربری، ابزار گرافیکی و غیره. بدین ترتیب شما می‌توانید بدون صرف انرژی بر روی اجزای سطح پایین که قبلاً طراحی شده‌اند، محصولات باارزشی تولید کنید.

تجاری‌سازی با یک مجوز دوگانه

مفهوم مجوز دوگانه را در فصل سوم به طور مختصر شرح دادم و فایل سیستم Reiser FS را به عنوان مثال آوردم. در این قسمت مجوز دوگانه به عنوان یک مدل تجاری بررسی خواهد شد. در این مدل خاص نیازمند در اختیار داشتن مجوز کپی، یا هر مجوز قابل قبول دیگری برای تمامی کدهای منبع هستیم. مجوز دوگانه بدین معناست که شما یک نسخه متن‌باز از نرم‌افزار را نگهداری کنید و همزمان یک نسخه از همان نرم‌افزار را تحت مجوزهای تجاری عرضه کنید. این مدل تجاری حول مجوزهای سنتی نرم‌افزار و درآمد ناشی از پشتیبانی نسخهٔ تجاری، می‌چرخد. این مدل اغلب تفاوتهایی بین نسخه‌های متن‌باز و تجاری نرم‌افزار قائل می‌شود. این تفاوتها اغلب به این صورت هستند که نسخه‌های جدید نرم‌افزارها که به صورت تجاری عرضه می‌شوند، کاملاً تست شده‌اند و اشکالات کمتری دارند و ممکن است امکاناتی در آنها باشد که در نسخهٔ متن‌باز نباشد. در این مدل یک پارامتر زمان هم وجود دارد. بدین معنا که نسخه‌های متن‌باز چندین ماه بعد از نسخه‌های تجاری عرضه می‌شوند. شکل ۱۱-۳ نمای کاملی است از این مدل.



شکل ۱۱-۳: مدیریت نرم‌افزار تجاری دارای مجوز دوگانه

در میان مزایای این مدل تجاری می‌توان به امکان بهره‌گیری از برنامه‌نویسانی که به نرم‌افزار شما علاقمند هستند، اشاره کرد. شما می‌توانید از این فرصت برای کاهش هزینه‌های برنامه‌نویسی استفاده کنید. نسخهٔ متن‌باز نرم‌افزار باعث ایجاد جمعیت بزرگ در کاربران علاقمند به این نرم‌افزار می‌شود و این به نوبهٔ خود تبلیغی است برای محصول شما. گرچه ممکن است نسخهٔ تجاری نرم‌افزار از نسخهٔ متن‌باز بهتر باشد، اما جامعهٔ متن‌باز را ارضاء می‌کند. گرچه در دسترس بودن نسخه متن‌باز باعث از دست دادن بخشی از درآمد این نرم‌افزار می‌شود، اما گسترش کاربران و آگاهی جامعه از امکانات این محصول، اغلب منجر به افزایش تعداد مشتریان شما می‌شود.

برای پیاده‌سازی این مدل باید تمامی یا بخشی از نرم‌افزار خود را تحت یکی از مجوزهای متن‌باز عرضه کنید. در فصل بعد شرایط عرضه محصول بررسی خواهد شد که یکی از آنها مالکیت حق کپی تمامی قسمتهای نرم‌افزار می‌باشد. شرکت شما باید وظیفهٔ نگهداری نسخهٔ متن‌باز نرم‌افزار را نیز به عهده گیرد. به عنوان نگهدارندهٔ نرم‌افزار، پذیرش یا رد اصلاحات عرضه شده بر عهده شماست. پس از تصمیم‌گیری در این زمینه، باید از اشخاصی که اصلاحات قابل قبولی ارائه کرده‌اند، درخواست کنید که حق کپی آن را به شما واگذار کنند. بنابراین حق کپی کارهای عرضه شده توسط دیگران نیز در اختیار شرکت شما خواهد بود. در ضمیمهٔ «ب» یک نمونه از نحوهٔ واگذاری حق کپی آمده است. به هر حال در تمامی مراحل باید از مشاوره یک حقوقدان استفاده کنید تا فعالیت‌های شما برخلاف قانون‌های محلی و جهانی نباشد. در واگذاری حق کپی اغلب این امکان به صاحب آن داده می‌شود که یک نرم‌افزار را هم تحت مجوزهای اختصاصی و هم تحت مجوزهای متن‌باز عرضه کند. اگر حق کپی نرم‌افزاری در انحصار شما باشد، می‌توانید آن را با هر نرم‌افزار اختصاصی دیگری که مالکیت آن را در اختیار دارید، ترکیب کنید.

سخت افزار

یکی از مدلهایی که به آسانی می‌توان پیاده‌سازی کرد، مدل تجاری سخت‌افزار می‌باشد. این مدل فرض می‌کند که ارزشی را که به مشتری ارائه می‌کنید، در سخت‌افزار است. سخت‌افزاری که

عرضه می‌کنید، معمولاً نوعی دستگاه جانبی یا سخت‌افزاری رابط برای یک دستگاه جانبی است. به عنوان مثالهایی از دستگاه جانبی می‌توان به پرینتر، اسکنر و دیسک‌گردان اشاره کرد. کارت‌های گرافیکی، کنترلرهای دیسک و رابط‌های USB نمونه‌هایی از رابط‌های سخت‌افزاری هستند.

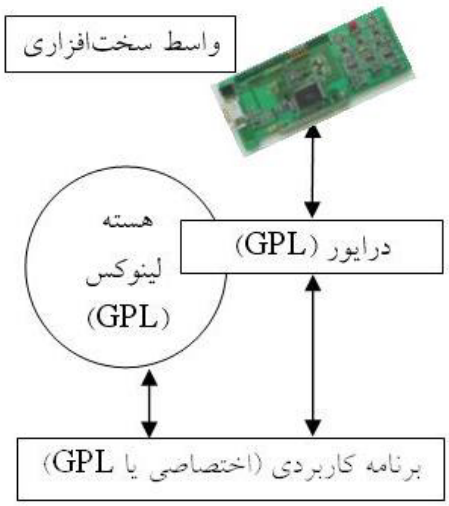
اگر هدف اولیه شما عرضه سخت‌افزار است باید کاربرد آن را به جاهای بیشتری گسترش دهید. یکی از راه‌های گسترش بازار با کمترین هزینه این است که از جامعهٔ متن‌باز برای نوشتن درایورهای سخت‌افزار خود (نرم‌افزار رابط بین سیستم‌عامل و سخت‌افزار) بهره گیرید.

تمایز بین دستگاه واسط و کارائی‌های آن

برای توضیح این مفاهیم، مثالی می‌زنم. فرض کنید شما رابط سخت‌افزاری می‌فروشید که امکان تماشای تلویزیون بر روی کامپیوتر را در اختیار مشتریان می‌گذارد. سخت‌افزار شما نوعی کارت واسط خواهد بود که داخل کامپیوتر نصب می‌شود و می‌تواند سیگنال تلویزیون را دریافت کند. در این مثال حداقل دو نرم‌افزار باید همراه این دستگاه عرضه شود. یکی نرم‌افزار درایور است که کارت واسط را کنترل می‌کند و دیگری نرم‌افزاری است برای نمایش تلویزیون. سؤالات مدل تجاری شما باید اینها باشند:

- آیا مشتری من ارزش نرم‌افزار درایور را می‌داند؟
- آیا مشتری من ارزش نرم‌افزار نمایشگر تلویزیون را می‌داند؟

برای اینکه بتوانید از جامعهٔ متن‌باز بهره گیرید و بازار خود را گسترش دهید باید به سؤال اول پاسخ منفی دهید. پاسخ سؤال دوم ممکن است منفی یا مثبت باشد. در شکل ۱۱-۴ ارتباط بین نرم‌افزار، درایور و سخت‌افزار واسط نمایش داده شده است.



شکل ۱۱-۴: واسط‌های سخت‌افزاری

با دادن پاسخ منفی به سؤال اول، شما تضمین می‌کنید که این سخت‌افزار در بسترهای مختلف قابل استفاده است و این موقعیت شما را در بازار بهبود می‌بخشد. بنابراین قدم بعدی شما باید این باشد که شراکت نزدیکی را با جامعهٔ متن‌باز آغاز کنید و آنها را تشویق به پیاده‌سازی سخت‌افزار شما در بسترهای سخت‌افزاری و نرم‌افزاری مختلف کنید. همان گونه که در فصل قبل اشاره شد باید حداقل منابعی را به این پروژه اختصاص دهید تا بتوانید آن را در جامعهٔ متن‌باز هدایت کنید. اگر با جامعهٔ متن‌باز همکاری کنید آنها اکثر کار برنامه‌نویسی را برای شما انجام خواهند داد. بدین ترتیب می‌توانید چرخهٔ توسعه را سریعتر کنید و مثلاً با دادن سخت‌افزار تولیدی خود به اعضای کلیدی این جامعه، ارتباط خود را با آنها مستحکم کنید. اگر هزینهٔ تولید سخت‌افزار شما چند صد دلار باشد، معاملهٔ بسیار خوبی انجام داده‌اید. در زمینهٔ میزان کنترلی که می‌خواهید بر روی نرم‌افزار درایور داشته باشید نیز باید از قبل تصمیم گرفته باشید. این تصمیم بستگی به این دارد که چگونه می‌خواهید این نرم‌افزار را پشتیبانی و نگهداری کنید. ممکن است توسعهٔ نرم‌افزار درایور را به یکی از برنامه‌نویسان کلیدی جامعهٔ متن‌باز واگذار کنید و از او به عنوان نگهدارندهٔ این درایور استفاده کنید. راه دیگر این است که نگهداری نرم‌افزار درایور را داخل شرکت انجام دهید. هر یک از این دو راه را می‌توانید انتخاب کنید، بسته به اینکه چقدر قصد دارید روی طراحی درایور سرمایه‌گذاری کنید و به چه میزان می‌خواهید بر روی آن کنترل داشته باشید. حتی اگر تصمیم به

نگهداری نرم‌افزار در داخل شرکت گرفتید، جامعهٔ متن‌باز این حق را دارد که نسخه‌های بهتری از این درایور را طراحی کند. ارتباط مستقیم با جامعه باعث می‌شود که هر دو طرف کار خود را به درستی انجام دهند و از کدنویسی‌های تکراری جلوگیری شود.

با اینکه تکلیف نرم‌افزار درایور روشن شد، محصول شما هنوز قابل عرضه به مشتریان نیست. آنچه که اکنون شما نیاز دارید، برنامه‌ای است که امکان دیدن تلویزیون را به کاربران بدهد. در این نرم‌افزار باید امکاناتی از قبیل عوض کردن کانال، تنظیم بلندی صدا و تمامی امکانات یک تلویزیون معمولی موجود باشد. نرم‌افزارهای پیشرفته‌تر ممکن است امکانات جالب‌تری از قبیل ضبط برنامه‌های تلویزیون، نمایش نام ایستگاه‌های تلویزیونی و رد کردن آگهی‌های بازرگانی داشته باشند. اگر سخت‌افزار درست عمل کند و نرم‌افزار درایور هم نوشته شده باشد، برنامه‌نویسان خلاق جامعهٔ متن‌باز بلافاصله راه حل‌های خود را ارائه خواهند داد. حتی اگر جامعه نرم‌افزارهای جذاب‌تری برای سخت‌افزارهای شما طراحی کند، هنوز این امکان برای شما وجود دارد که قابلیتهای بهتری را به آنها اضافه کنید و آنها را به صورت نرم‌افزار اختصاصی و پشتیبانی شده به فروش رسانید. این تصمیم بستگی به توانایی شما در جذب درآمد بیشتر از محصولات اختصاصی و میزان تمایل شما به عرضه پشتیبانی این محصول دارد. شکل ۱۱-۴ نشان می‌دهد که چگونه ممکن است درایور GPL باشد، ولی کاراییهای باارزش‌تر در قالب نرم‌افزار نمایش تلویزیون، توسط جامعهٔ متن‌باز طراحی شود و یا به صورت نرم‌افزاری اختصاصی عرضه شود.

مستندات باز برای واسط سخت‌افزاری

برای اینکه جامعهٔ برنامه‌نویسان در طراحی نرم‌افزار درایور موفق شود، باید مستندات کافی در زمینهٔ عملکرد سخت‌افزار را در اختیار آنها قرار داد. گرچه نمونه‌های زیادی از مهندسی معکوس در جامعهٔ متن‌باز انجام شده است، اما اگر قرار است آنها درایور شما را بنویسند، منطقی نیست که اطلاعات مربوط به سخت‌افزار را از آنها مخفی کنید. منتشر کردن اسناد قابلیتهای داخلی یک

محصول برای بسیاری از مدیران قابل قبول نیست، اما اغلب لازم نیست تمامی مستندات سخت‌افزار را منتشر کنید. آنچه که باید منتشر شود فقط اسناد مربوط به کنترل سخت‌افزار است.

بهترین راه ایجاد این مستندات این است که گروهی از مهندسان آن را به کمک مستندات اصلی سخت‌افزار تهیه کنند. این مستندات فقط نحوهٔ ارتباط با سخت‌افزار را شرح می‌دهد، بدون اینکه اطلاعات خصوصی آن را که به درد رقبا می‌خورد، آشکار کند.

در مواردی که محصول شما تحت مجوز شرکت دیگری تولید می‌شود، قبل از انتشار این مشخصات باید با تولیدکننده اصلی سخت‌افزار (OEM) مذاکره کنید. در چنین مواردی ممکن است خود تولیدکننده اصلی درایور را برای شما طراحی کند.

اگر به هر دلیلی نمی‌خواهید هیچ یک از مستندات را منتشر کنید، می‌توانید یک نرم‌افزار درایور مرجع را طراحی کنید و آن را در اختیار جامعهٔ متن‌باز قرار دهید. این جامعه می‌تواند از این مدل مرجع به عنوان نقطهٔ شروع و جایگزینی برای مستندات سخت‌افزار استفاده کند.

همراه کردن نرم‌افزارهای مورد نیاز با سخت‌افزار

قدم آخر مدل تجاری سخت‌افزار این است که نرم‌افزارهای متن‌باز را به همراه سخت‌افزار خود عرضه کنید. به یاد داشته باشید هدف شما این است که ارزش بیشتری را به مشتریان عرضه کنید. در استفاده از یک نرم‌افزار متن‌باز، لزوم پشتیبانی از آن را نیز در نظر داشته باشد.

مدل پایان چرخهٔ زندگی

این مدل نیازمند تحلیل پیچیده‌ایست که بتواند اهداف کاهش هزینه، و سرعت بخشیدن به پایان زندگی و ایجاد یک جامعهٔ پویا را متعادل و برآورده کند.

وقتی که شرکتی عمر یک محصول را به پایان می‌رساند، باید تأثیرات آن را بر مشتریان در نظر داشته باشیم. مشتریانی که روی محصولی سرمایه‌گذاری کرده‌اند، نسبت به بی‌ارزش شدن سرمایه خود بی‌تفاوت نخواهند بود. آنچه که شرکت‌ها در این شرایط باید در نظر داشته باشند، تأثیر

منفی‌ای است که این کار بر وجهه آنها نزد مشتریان می‌گذارد. گرچه اغلب مشتریان منطق پشت پایان دادن به زندگی یک محصول را درک می‌کنند، اما آنها همچنان بخاطر هزینه‌هایی که صرف تهیه آن کرده‌اند، عصبانی خواهند شد.

اگر تولید یک محصول را متوقف می‌کنید، باید پشتیبانی آن را تا چندین سال ادامه دهید تا مشتریان ناراضی نشوند و زمان کافی برای مهاجرت به محصولات جدید را داشته باشند. در مدت پشتیبانی باید تعدادی از مهندسان را مأمور پاسخگویی به این مشتریان و رفع مشکلات آنها کنید. گرچه مشتریان حاضرند هزینه این پشتیبانی‌ها را پرداخت کنند، اما همچنان بخشی از منابع و انرژی شما بر چیزی متمرکز شده است که جزء اهداف شما نیست.

در چنین شرایطی، متن‌باز بهترین وسیله برای حفظ وجهه نزد مشتریان و پایین بردن هزینه‌های پشتیبانی می‌باشد. بدین ترتیب، می‌توانید تمام انرژی خود را بر اهداف درآمدزای شرکت متمرکز کنید. اگر در مدل تجاری شما اصولاً پشتیبانی در نظر گرفته نشده است، با از طریق متن‌باز کردن، می‌توانید این وظیفه را بر عهدهٔ شرکت‌های دیگری که متخصص در زمینهٔ پشتیبانی هستند، واگذار کنید. همچنین می‌توان محیطی ایجاد کرد که خود مشتریان پشتیبانی و بهبود آن محصول را ادامه دهند.

اگر قصد دارید کسب و کار کنونی خود را کلاً ترک کنید و در زمینهٔ دیگری آغاز به کار کنید، در این صورت محصول جدیدی نیز وجود نخواهد داشت که مشتریان به فکر مهاجرت به آن باشند. اما در هر کسب و کاری باید وجهه خود را نزد مشتریان و صنعت حفظ کنید و آن را از اولویت‌های خود قرار دهید. در چنین شرایطی بهترین کار این است که نرم‌افزار خود را تحت مجوزهای متن‌باز در اختیار عموم قرار دهید.

ایجاد یک اکوسیستم

در فصل قبل شبیه‌ساز Palm OS را به شما معرفی کردیم. این نرم‌افزار یکی از اولین نمونه‌های موفق مدلهای تجاری متن‌باز است. در این مثال نشان داده شد که چگونه یک شرکتی

ارزش واقعی خود را تشخیص داد و در ارتباط نزدیکی با جامعهٔ متن‌باز کار کرد. در اینجا بررسی تاریخچه شبیه‌ساز Palm OS خالی از لطف نیست.

وقتی که US Robotics خانوادهٔ Pilot را به بازار عرضه کرد، بستر برنامه‌نویسی شرکتی آن نرم‌افزاری بود از شرکت Metrowerks. شرکت Metrowerks در طراحی ابزار برنامه‌نویسی برای محیطهای Embedded تخصص داشت. برای برنامه‌نویسان تجاری این ابزار قابل قبول بود، اما اکثر برنامه‌نویسان ابزار برنامه‌نویسی می‌خواستند که تحت پلاتفرم PC نیز قابل اجرا باشد. برنامه‌نویسان وقتی که برای دستگاه‌های Embedded مانند PDAها برنامه‌نویسی می‌کنند، ترجیح می‌دهند برنامهٔ خود را در همان سیستمی که برنامه‌نویسی روی آن صورت می‌گیرد، آزمایش کنند. بنابراین باید امکان شبیه‌سازی PDA در محیط دیگری مانند PC باشد. این شبیه‌سازی علاوه بر اینکه آزمایش نرم‌افزار را راحت‌تر می‌کند، خلاقیت و سرعت برنامه‌نویس را نیز بالا می‌برد. بدون شبیه‌ساز، برنامه‌نویس باید پس از هر تغییری برنامه را به یک PDA واقعی برای آزمایش منتقل کند. قابل ذکر است که در PDAها به دلیل محدود بودن امکانات، برنامه‌نویسی بسیار مشکل است. بنابراین باید این کار در سیستمهای دیگری مانند PCها صورت گیرد. گرچه ابزار برنامه‌نویسی Metrowerks بسیار کامل و جامع بود، اما آنقدر گران بود که افراد عادی قادر به خرید آن نبودند. از طرف دیگر ابزار این شرکت فقط تحت سیستمهای مکینتاش قابل استفاده بود. از آنجایی که مکینتاش و پایلوت از نسل مشترکی از پردازنده‌ها (موتورولا ۶۸۰۰۰) استفاده می‌کردند، شبیه‌سازی پایلوت در مکینتاش بسیار راحت بود. به تدریج جامعه‌ای از برنامه‌نویسان پایلوت (امروزه Palm نامیده می‌شود) تشکیل شد. هدف آنها این بود که ابزار برنامه‌نویسی برای پایلوت بسازند که به راحتی در پلاتفرم PC قابل استفاده باشد. در آن زمان مفاهیم متن‌باز به این گستردگی مطرح نشده بود و بسیاری از آن اطلاع نداشتند. این برنامه‌نویسان مجبور بودند تمامی اجزای شبیه‌ساز مورد نظر را به کمک مهندسی معکوس بسازند. این اجزا یکی پس از دیگری کنار یکدیگر قرار گرفتند. یکی از برنامه‌نویسان توانست فرمت برنامه‌های اجرایی Palm را بدست آورد. برنامه‌نویس دیگری ابزاری

برای کامپایل کردن منابع رابط کاربری نوشت. برنامه‌نویس دیگری هم کمپایلرهای متن‌باز را برای ساختن (کامپایل) و غلط‌یابی برنامه‌های Palm اصلاح کرد. در همین حال، گِرگ هیوجیل سرگرم بررسی شبیه‌سازهای Pilot تحت پلاتفرم‌های دیگر بود و سرانجام CoPilot نوشته شد. گرچه این وقایع بسیار ساده به نظر می‌رسند، اما همین وقایع ساده منجر به تولید کاملترین شبیه‌ساز Pilot تحت PC شدند. این شبیه‌ساز گرچه به زیبایی نسخه Metrowerks نبود، اما از نظر امکانات چیزی از آن کمتر نداشت.

مشکل دیگری که گِرگ باید حل می‌کرد مسئله ROM دستگاه بود. در PDAها، سیستم‌عامل در حافظهٔ فقط خواندنی یا ROM قرار دارد. آنچه که گِرگ طراحی کرده بود، فقط یک سخت‌افزار PDA را شبیه‌سازی می‌کرد، اما برای اینکه بتوان از آن شبیه‌سازی استفاده کرد، باید سیستم‌عامل PDA نیز به آن اضافه می‌شد. او توانست به کمک سایر برنامه‌نویسان، ابزاری برای خواندن ROM دستگاه بنویسد. به کمک این ابزار می‌توانستند ROM دستگاه را به صورت قطعه‌هایی دریافت و به شبیه‌ساز اضافه کنند (در نظر داشته باشید که امروزه قوانینی وجود دارد که اجازهٔ این کار را به شما نمی‌دهد). بدین ترتیب پلاتفرم شبیه‌سازی شده کامل شد. اما از آنجایی که USRobotics مجوز انتشار کد ROM را نداده بود، هر کاربری باید ROM دستگاه خود را استخراج و در شبیه‌ساز استفاده می‌کرد. این بدین معنا بود که اگر قصد نوشتن برنامه‌ای برای یک PDA داشتید، باید یک دستگاه از آن را تهیه می‌کردید.

از اینجا نقطهٔ پویایی Pilot آغاز شد. تعداد برنامه‌های Pilot به طور غیرقابل تصوری افزایش یافت. تعداد برنامه‌نویسان این دستگاه نیز افزایش یافت و یک اکوسیستم متکی به خود ایجاد شد. در آن زمان USRobotics (و سپس 3Com) از درک شرایط پیش آمده عاجز بودند. از یک طرف محرمانگی اطلاعات اختصاصی دستگاه از بین رفته بود و از طرف دیگر این جامعه برنامه‌نویسان با نوشتن برنامه‌های متنوع، ارزش این دستگاه را بالاتر برده بودند. وقتی که Palm از این شرکت‌ها مستقل شد، در آستانه یک تصمیم مهم قرار گرفت: آیا از این جامعه استقبال کند یا با آن بجنگد و

کنترل PDA را دوباره بدست آورد؟ Palm تصمیم به استقبال از این جامعه گرفت. اکنون Metrowerks ابزار برنامه‌نویسی خود را هم برای مکینتاش و هم برای PC عرضه کرده است. تولیدکنندگان نرم‌افزار تجاری همچنان از این ابزار برای نوشتن برنامه‌های این دستگاه استفاده می‌کند. اما افراد عادی ابزار دیگری در اختیار دارند که رایگان است. Palm و Metrowerks ارزش CoPilot را درک کردند و مجوز انتشار آن را از گِرگ و سایر برنامه‌نویسان گرفتند. سپس نام آن را به شبیه‌ساز PalmOS (POSE) تغییر دادند و تحت مجوز GPL منتشر کردند. امروزه POSE را Metrowerks به همراه نسخهٔ تحت PC ابزار برنامه‌نویسی خود عرضه می‌کند. این برنامه از سایت Palm نیز به طور رایگان قابل دریافت است.

شرکت Palm به این نتیجه درست رسید که POSE ابزار با ارزشی است که مجموعه ابزار برنامه‌نویسی Palm را تکمیل می‌کند. دقت کنید که صِرف عرضه این ابزار به مشتریان ارزشی ندارد. بلکه ارزش در کامپیوتری قابل حمل نهفته است که طیف گسترده‌ای از نرم‌افزارهای کاربردی برای آن وجود دارد. یعنی ارزش این دستگاه با افزایش تعداد برنامه‌ها بالا می‌رود و POSE این مهم را تحقق بخشیده است.

امروزه شرکت Palm و جامعهٔ برنامه‌نویسان به طور مشترک روی POSE کار می‌کنند. Palm وظیفهٔ نگهداری این برنامه را بر عهدهٔ عده‌ای از برنامه‌نویس خود قرار داده است. اما جامعه برنامه نویسان، آن را بهبود می‌دهد و اشکالات آن را برطرف می‌کند. بدین ترتیب Palm مسیر توسعهٔ POSE را در جهت بالابردن ارزش سخت‌افزارهایش هدایت می‌کند. بدین ترتیب این شرکت توانسته است هزینه‌های خود را در سطح پایینی نگه دارد و اکوسیستم باارزشی ایجاد کرده است. امروزه Palm برنامه موجود در ROM هر دستگاه را آزادانه در اختیار عموم قرار می‌دهد و نسخه‌های جدیدتر سیستم‌عامل را در سایت خود می‌گذارد.

آنچه که دیدیم، مثال بسیار خوبی بود از یک مدل تجاری موفق. در این مدل تجاری نشان داده شد برای بهره برداری از متن‌باز، باید بدانیم ارزش واقعی محصول ما در کجای آن قرار دارد. نکات قابل توجه در این مدل تجاری عبارتند از:

- شرکت Palm به درستی ارزش آنچه را که به مشتریان عرضه می‌کرد، فهمید: یک دستگاه قابل حمل با تعداد بسیار زیادی برنامه.
- جامعه برنامه‌نویسان و Palm اکوسیستم با ارزشی درست کردند که باعث ترویج Palm و بالا رفتن ارزش آن شد.
- جامعهٔ برنامه‌نویسان یکی از ابزارهای کلیدی زنجیرهٔ ارزش Palm را طراحی کرد که به برنامه‌نویسان این امکان را می‌داد که برنامه‌های Palm را در یک PC بنویسند و آزمایش کنند.
- Palm همچنان در توسعهٔ POSE با این جامعه همکاری می‌کند. این همکاری باعث می‌شود شبیه‌ساز همان مسیری را طی کند که Palm می‌خواهد و در عین حال دانش برنامه‌نویسان در زمینهٔ این دستگاه نیز بالا می‌رود.
- عرضه کردن برنامه موجود در ROM نوعی از عرضهٔ مستنداتی است که جامعه برنامه‌نویسان برای موفقیت در کار خود، به آن نیاز دارند. البته این شرکت مستندات دیگری نیز در اختیار برنامه‌نویسان قرار داده است.

هیچ کس در USRobotics ، 3COM و Palm تصور چنین وقایعی را نمی‌کرد. اکنون شما این شانس را دارید که از آنها که این مسیر را طی کرده‌اند یاد بگیرید و آموخته‌های خود را در سازمان خود اعمال کنید.

جمع بندی

شکست یا موفقیت یک مدل تجاری بستگی به این دارد که چگونه اجرا شود. فهمیدن یک مدل تجاری جای تحلیل جامع تجاری را نمی‌گیرد. شما هنوز هم باید بازار و نیازهای مشتریان خود

را بشناسید و همهٔ عناصر مورد نیاز برای ایجاد یک کسب و کار سودده را نیاز دارید. متن باز فقط یکی از این عناصر است که دروازه‌هایی را برای ساختن یک مدل تجاری جدید یا بهبود مدل کنونی به روی شما می‌گشاید.

اکنون که شما طرح‌های تجاری قابل استفاده را به خوبی درک کرده‌اید، باید نحوهٔ پیاده‌سازی متن‌باز را در سازمان خود بیاموزید.

در فصل بعد در مورد تصمیم‌گیری در زمینهٔ استفاده و عرضهٔ محصولات متن‌باز بحث خواهد شد.

فصل ۱۲

پیاده‌سازی متن‌باز در کسب و کار

در سه فصل گذشته جزئیات نحوهٔ کار متن‌باز شرح داده شد و همچنین تأثیرات اقتصادی آن بر صنعت نرم‌افزار بررسی شد. در همین راستا چندین مدل تجاری برای شرکت‌های تولیدکنندهٔ نرم‌افزار معرفی شد. با فرض اینکه شما قصد پیاده‌سازی یکی از این مدلهای تجاری نوین را دارید، باید تغییرات پایه‌ای مورد نیاز در کسب و کار خود را معین کنید. در این فصل و فصل بعد فرآیندهای مورد نیاز برای تصمیم‌گیری درست در این زمینه شرح داده خواهد شد. اهداف این فصل عبارتند از:

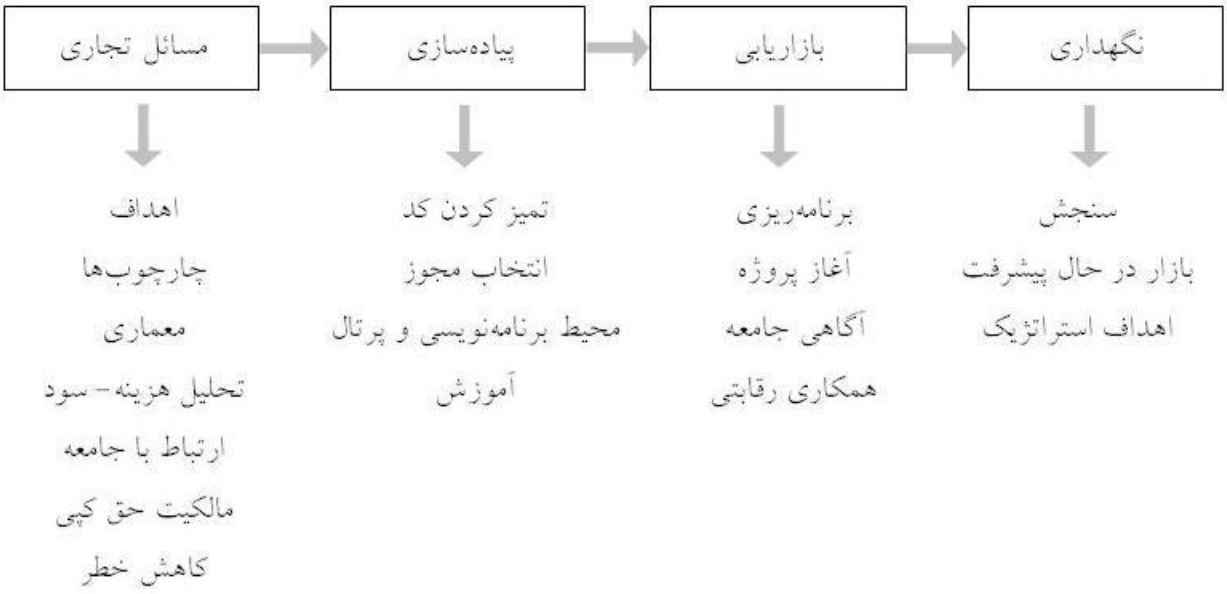
- بررسی دقیق دلایل متن‌باز کردن یک نرم‌افزار
- تهیه لیستی از کارهایی که موقع متن‌باز کردن یک نرم‌افزار اختصاصی باید انجام داد
- بررسی مشکلات به کارگیری متن‌باز در کسب و کار
- بررسی مسائل قانونی استفاده از متن‌باز

این فصل هم برای مدیران فناوری اطلاعات و هم برای شرکت‌های تولیدکنندهٔ نرم‌افزار مفید است. مدیران فناوری اطلاعات دوست دارند از نرم‌افزارهای متن‌باز به عنوان راهی برای کاهش هزینه‌ها استفاده کنند. اما برای شرکت‌های تولیدکننده نرم‌افزار هر دو مقولهٔ آوردن نرم‌افزار به داخل شرکت و عرضهٔ نرم‌افزار متن‌باز قابل توجه خواهد بود.

عرضهٔ نرم‌افزار متن‌باز

منظور از عرضهٔ نرم‌افزار متن‌باز در اینجا این است که نرم‌افزاری که در حال حاضر به صورت اختصاصی به فروش می‌رود، تحت یک مجوز متن‌باز عرضه شود. برای انجام موفق آمیز این کار، باید مراحل نمایش داده شده در شکل ۱۲-۲ را دنبال کنید.

همان گونه که در شکل ۱۲-۲ می‌بینید، عرضهٔ نرم‌افزار به جامعهٔ متن‌باز فرآیند آسانی نیست. برای انجام این کار باید برنامه‌ریزی و تحلیل دقیقی صورت گیرد. البته پروژه‌ها متفاوت هستند و میزان پیچیدگی آنها با یکدیگر فرق می‌کند. در برخی از موارد کل پروسه ممکن است در مدت کوتاهی انجام شود. اما به هر حال باید حداقل تأملی در عناصر دخیل در این فرآیند صورت گیرد. یک خلبان هر چقدر هم باتجربه باشد، یکسری موارد را قبل از پرواز چک می‌کند.



شکل ۱۲-۱: فرآیند متن باز کردن یک نرم‌افزار

مسائل تجاری

اولین قدمی که باید برداشته شود این است که مسائل تجاری این فرآیند دقیقاً بررسی شود و این مسئله مشخص شود که آیا انجام این کار منطقی است یا نه. در شکل ۱۲-۱ عناصر کلیدی

مربوط به مسائل تجاری مشخص شده‌اند. از آنجایی که شما در دنبال کردن این فرآیند مبتدی هستید بررسی هر یک از این عناصر خالی از لطف نیست.

اهداف

قبل از آغاز هرگونه پیاده‌سازی باید دلایل منطقی برای آغاز پروژه‌ها به صورت متن‌باز داشته باشیم و همهٔ افراد درگیر در آن پروژه از آن اهداف به خوبی آگاه باشند. در اینجا چند مثال از اهداف مختلف یک کسب و کار آمده است. اکثر این مثالها با مدل تجاری مطرح شده در فصل قبل همخوانی دارد.

- **بی‌ارزش کردن یک محصول رقابتی**- اگر رقیب شما محصولی داشته باشد که از محصول شما بسیار بهتر است یا شما اصلاً آن را عرضه نمی‌کنید، انگیزه کافی برای استفاده از این روش را دارید. بنابراین هدف شما این است که سرعت کاهش ارزش محصول رقیب را افزایش دهید و جریان درآمد و سود او را قطع کنید. اما این نکته را در نظر داشته باشید که با انجام این کار بر تمام بازار که خود شما هم عضوی از آن هستید، تأثیر می‌گذارید.
- **قدرت نفوذ**- اگر فناوری در انحصار شما راهی برای ارتقاء محصولات دیگران است، می‌توانید از متن باز کردن به عنوان روشی برای مطرح شدن خود و انتخاب آن فناوری به عنوان استاندارد استفاده کنید.
- **کاهش هزینه‌ها**- در شرایط خاص، کار مشترک با جامعهٔ متن‌باز هزینه‌های تولید نرم‌افزار را به طور قابل توجهی کاهش می‌دهد. در چنین مواردی اگر در حال عرضهٔ محصولی بزرگ هستید، متن‌باز کردن برخی از اجزاء آن سرعت توسعهٔ آن را بیشتر و هزینهٔ آن را کمتر می‌کند.
- **سخت‌افزار**- همان گونه که در مدل تجاری سخت‌افزار اشاره کردم، اگر سخت‌افزار خاصی را عرضه می‌کنید، در بهترین شرایط برای کار با جامعهٔ متن‌باز هستید.

- **خدمات و پشتیبانی-** اگر مدل تجاری شما بر اساس ارائه خدمات بنا شده است، توصیه محصولات نرم‌افزاری دیگران به مشتریان، منطقی به نظر می‌رسد.
- **طرح خروج-** اگر در حال ترک کسب و کار کنونی خود هستید، متن‌باز کردن محصولات و مستندات می‌تواند بسیار مفید باشد.
- **شراکت-** متن‌باز تشویق به شراکت و همکاری می‌کند. شما می‌توانید از متن‌باز به عنوان راهی برای کار کردن با شرکا و رقبا بر روی پروژه‌های سنگین، استفاده کنید.
- **انتشار فناوری از رده خارج-** اگر شرکت شما فناوری را توسعه داده است که امروزه جزو اهداف استراتژیک آن نیست و قصد ندارید هزینهٔ ثبت آن را پرداخت کنید، می‌توانید آن را تحت مجوزهای متن‌باز منتشر کنید.

به همین ترتیب دلایل زیادی هم وجود دارد برای اینکه متن‌باز کردن به سوددهی نمی‌رسد. تعدادی از این دلایل عبارتند از:

- **محصول یکه تاز-** اگر محصول شما حرف اول را در میان محصولات موجود می‌زند و مانع ورود محکمی برای رقباست، نباید آن را متن‌باز کنید.
- **محصول منسوخ شده-** پایان عمر یک محصول لزوماً به معنای منسوخ شدن آن نیست. اگر احساس می‌کنید که محصول شما از دید شما و مشتریانتان دیگر قابل استفاده نیست، بلافاصله عرضهٔ آن را متوقف کنید. در چنین شرایطی، زنده نگهداشتن محصول از طریق متن‌باز کردن، به هیچ وجه توصیه نمی‌شود.
- **سود منفی-** اگر بررسی‌های شما نشان دهد که سود متن‌باز از ضررهای آن کمتر است، نباید به سراغ آن بروید.
- **عدم تمرکز منابع-** اغلب پس از عرضهٔ محصول به جامعهٔ متن‌باز نیز مجبور به صرف قسمتی از منابع داخلی خود برای آن هستید. اگر این کار باعث عدم تمرکز منابع سازمان شما و هدف اصلی شود، باید از آن اجتناب کنید.

- **خطرات مالکیت معنوی**- وکیل شرکت باید مدتی با شما کار کند تا بفهمد چه کسی مالکیت معنوی هر محصول را در اختیار دارد. اگر خطرات قانونی مالکیت معنوی زیاد باشد، در این صورت نباید این گونه پروژه‌ها را ادامه دهید.
- **رقابت با جامعه**- رقابت با پروژه متن‌باز موجود به هیچ وجه توصیه نمی‌شود و فقط وجهه شما را در جامعهٔ متن‌باز خراب می‌کند. اگر پروژه‌ای مشابه پروژه شما در این جامعه در حال توسعه است، باید به جای رقابت با آن، تلاش کنید که به آن بپیوندید و در بهبود آن تلاش کنید. البته در مواردی که پروژه‌ای اهدافی غیر از اهداف شما را دنبال می‌کند، رقابت با آن مشکلی ندارد. برای اینکه درست تصمیم بگیرید، باید از راهنمایی‌های جامعهٔ متن‌باز استفاده کنید.
- **فناوری جالب**- بسیاری از مهندسان فکر می‌کنند چون فناوری جالبی طراحی کرده‌اند، بلافاصله باید آن را متن‌باز کنند. دیدگاه آنها این است که تشویق‌های مردم باعث پیشرفت بهتر پروژه می‌شود. اما در اغلب موارد هزینه سنگین پروژه بر تشویق‌های زودگذر غالب می‌شود و اگر پروژه به نتیجه مطلوب نرسد، واکنشهای منفی، به موارد فوق افزوده می‌شود.

وقتی که اهداف کلیدی شرکت شما برای یک پروژه مشخص شد، می‌توانید قدم بعدی را در زمینهٔ مسائل تجاری بردارید.

معیارها

پس از تعیین اهداف پروژه، باید معیارها را مشخص کنیم. این معیارها برای سنجش میزان موفقیت پروژه لازمند. تعدادی از معیارها عبارتند از:

- رقیبی که باعث کاهش قیمت محصول شده است
- تعداد برنامه‌نویسانی که به طور فعال در پروژه نقش دارند
- تعداد کارهای ارائه شده در پروژه

- تعداد کاربرانی که از پروژه استفاده می‌کنند
- عناصری از اکوسیستم که حول پروژه شما در حال شکل گرفتن هستند
- کاهش هزینه‌های توسعه به کمک عرضه به موقع محصول

این معیارها بستگی به مشخصات پروژه دارند. شما باید بتوانید نقطهٔ موفقیت را تعریف و زمان رسیدن به آن را تعیین کنید.

معماری

در فاز معماری از منابع فنی‌ای که در پروسهٔ تصمیم‌گیری دخیل بودند، استفاده می‌شود. متخصصان فناوری باید معماری پروژه متن‌باز پیشنهاد شده را معین کنند. آنچه که تیم فنی باید مشخص کند، عبارتند از:

- آیا ماجول‌های مختلف نرم‌افزار به خوبی از یکدیگر مجزا شده‌اند؟
- آیا می‌توان نرم‌افزار را به اجزای متن‌باز و اختصاصی تقسیم کرد؟
- آیا این پروژه متن‌باز می‌خواهد پلی برای پروژه‌های اختصاصی باشد؟
- آیا رابط‌های مناسب برای اتصال این دو پروژه در نظر گرفته شده است؟
- آیا می‌توان این نرم‌افزار را از سیستمهای مدیریتی کنونی شرکت ایزوله کرد؟
- آیا مالکیت معنوی تمامی نرم‌افزار حاصل در اختیار شماست؟ در غیر این صورت آیا شما مجوز انتشار آنها را تحت مجوزهای متن‌باز دارید؟ آیا می‌توانید عناصر مشکل‌دار نرم‌افزار را عوض کنید؟
- آیا کدی که قرار است عرضه شود، قبلاً به نام کسی ثبت شده است؟ حتی اگر این کد به نام شما و شخص دیگری ثبت شده باشد، ممکن است شما اجازه انتشار آن را تحت مجوزهای متن‌باز نداشته باشید.

پروژه‌ها گوناگون هستند و هر نرم‌افزار معماری منحصر به فردی دارد. اگر اهداف پروژه را مشخص کرده باشید، تیم فنی می‌تواند بر این اساس، محصول قابل قبولی برای شما آماده کند.

تحلیل هزینه و سود

بسیاری از مردم فکر می‌کنند فرآیند انتقال یک نرم‌افزار به مجوزهای متن‌باز، فقط مستلزم بسته‌بندی آن نرم‌افزار و قرار دادن آن در اینترنت برای عموم است.

اما واقعیت این است که فرآیند متن‌باز کردن می‌تواند بسیار پرهزینه باشد. در هر تحلیل مالی باید اهداف و پویایی پروژه در نظر گرفته شود. برخی از مواردی که باید در تحلیل هزینه پروژه در نظر گرفته شود، عبارتند از:

- ایجاد یک مخزن نرم‌افزار جدید
- تصفیه کدهای موجود و حذف بخشهای اضافه و توضیحات برنامه (COMMENT)
- برنامه‌های تبلیغاتی برای اعلام متن‌باز شدن پروژه و ایجاد جامعه‌ای از برنامه‌نویسان حول آن
- نگهداری نرم‌افزار (با فرض اینکه شما می‌خواهید همچنان نگهدارنده آن نرم‌افزار باشید)
- انتشار مستندات: شامل معماری، طراحی و راهنمای کاربر
- ایجاد پرتال برای آن نرم‌افزار برای همکاری بهتر جامعه با پروژه
- آموزش نرم‌افزار: هزینه آموزش نرم‌افزار بستگی به میزان پیچیدگی آن دارد
- تحمل هزینه دریافت محصول از اینترنت: برای اینکه نرم‌افزار را در اختیار عموم قرار دهید، باید از کارگزارهایی برای دریافت آن استفاده کنید و هر چه میزان دریافت یک پروژه از اینترنت بیشتر باشد، شما باید برای مهار ترافیک ایجاد شده سرمایه‌گذاری کنید.

طبیعتاً انتظار دارید که این هزینه‌ها را درآمدی جبران کند. این درآمد ممکن است به صورتهای زیر باشد:

- کاهش هزینه‌های برنامه‌نویسی داخل شرکت
- کاهش هزینه‌های پشتیبانی

- شناخته شدن در بازار
- افزایش درآمد محصولات همراه (سخت‌افزاری یا نرم‌افزاری)
- ضرر کردن رقبا

این تحلیل هزینه و سود، اطلاعات کافی برای تصمیم گیری درست را در اختیار شما قرار می‌دهد.

ارتباط با جامعه

این یکی از مهمترین بخشهای فرآیند تصمیم‌گیری است که متأسفانه اغلب نادیده گرفته می‌شود. با اینکه بسیاری از حامیان متن‌باز معتقدند که همهٔ نرم‌افزارها باید در اختیار عموم قرار گیرند، همان گونه که دیدید این کار واقعاً هزینه‌بر است. فقط در صورتی متن‌باز کردن پروژه منطقی است که بتوان هزینه‌های آن را از جای دیگری جبران کرد. ممکن است نرم‌افزار شما به گونه‌ای باشد که متن‌باز کردن آن توجه جامعه را به خود جلب نکند. در این صورت این اقدام شما برای هیچ کس اهمیت نخواهد داشت و هیچ کس در توسعهٔ آن شرکت نخواهد کرد.

شما باید احتمال ایجاد جامعه برنامه‌نویسان برای پروژه آتی خود را پیش‌بینی کنید. اگر در حال افزودن عملکردی به یک پروژه موجود هستید، می‌توانید با سردمداران جامعهٔ برنامه‌نویسان درباره این پروژه صحبت کنید و نظر آنها را در زمینهٔ میزان تمایل افراد جامعه به شرکت در این پروژه جدید جویا شوید.

اگر قصد دارید این جامعه را به کمک مشتریان خود تشکیل دهید، باید از آنها نظرسنجی کنید تا از میزان تمایل آنها به شرکت در این جامعه برنامه‌نویسان آگاه شوید.

در نهایت اگر قصد دارید جامعهٔ جدیدی برای یک فناوری جدید ایجاد کنید باید در زمینهٔ شناساندن آن پروژه به برنامه‌نویسان تلاش کنید و در عین حال نباید هزینه‌های اجرای چنین پروژه‌ای را نیز دست کم بگیرید. برای جذب برنامه‌نویسان خبره باید در فکر ایجاد انگیزه در آنها نیز باشیم.

مالکیت حق کپی

گرچه من قبلاً هم به مسأله مالکیت حق کپی اشاره کرده‌ام، لازم به یادآوریست که این مسأله مهمترین بخش از فرآیند تصمیم‌گیری است. نه تنها مالکیت کدهای موجود را باید در نظر بگیرید، بلکه گرفتن مجوز کپی از کارهای ارائه شده توسط جامعه نیز حائز اهمیت است.

فرآیند تصمیم‌گیری تا حدی هم بستگی به مدل تجاری دارد که انتخاب کرده‌اید. اگر از فصل قبل به یاد داشته باشید، شما می‌توانید نرم‌افزار خود را تحت دو مجوز عرضه کنید: یکی متن‌باز و دیگری اختصاصی. اگر مدل تجاری مجوز دوگانه را انتخاب کرده‌اید، باید مراحلی را برای دریافت کارهای انجام شده از کسانی که قصد دارند حق کپی کارشان را به شما واگذار کنند، طراحی کنید یا اینکه مجوزی برای آن در نظر گیرید که امکان انشعاب به مجوزهای فرعی برنامه‌نویسان را داشته باشید.

کاهش خطر

مسلماً شما دوست دارید پروژه را به بهترین نحو به انجام برسانید، اما جنبه‌های منفی اجرای پروژه را نیز باید در نظر داشته باشید. باید احتمال درست پیش نرفتن کارها را بدهید و برای چنین شرایطی آمادگی کافی داشته باشید. در طرح کاهش خطرات باید موارد زیر را در نظر داشته باشیم:

- اگر جامعه برنامه‌نویسی حول پروژه شما ایجاد نشد، چه می‌کنید؟ چقدر منتظر ایجاد این جامعه می‌شوید؟ اگر جامعه علاقه‌ای برای شرکت در پروژه نشان نداد، آیا باز هم برای تبلیغ پروژه سرمایه‌گذاری می‌کنید؟ بر اساس چه معیاری ممکن است پروژه را متوقف کنید؟
- اگر هزینه‌های ÷روژه از ارقام پیش‌بینی شده تجاوز کرد، چه می‌کنید؟ آیا برنامه‌ای برای جبران این هزینه‌ها دارید؟
- اگر کسی در زمینهٔ مالکیت معنوی پروژه ادعا کرد، چه می‌کنید؟ آیا اسناد کافی در این زمینه در دست دارید؟

- اگر پروژه دیگری در رقابت با پروژه شما آغاز به کار کرد چه می‌کنید؟ آیا تمهیداتی برای جلوگیری از اجرای پروژه مشابه اندیشیده‌اید؟
- اگر جامعه متن‌باز شروع به اجرای پروژه مشابهی کرد که اهداف شما را تحت‌الشعاع قرار می‌دهند، چه می‌کنید؟ واکنش شما چه خواهد بود؟ آیا اصلاً اهمیتی برای شما دارد؟
- اگر میزان دریافت نرم‌افزار از سایت شما چنان ترافیک را بالا ببرد که عملاً دریافت آن امکانپذیر نباشد، چه می‌کنید؟ چه تمهیداتی برای بالابردن پهنای باند سایت در چنین شرایطی اندیشیده‌اید؟

در این زمینه وقت زیادی بگذارید و هر آنچه که ممکن است مشکل‌ساز شود، پیش‌بینی کنید. برای این کار، لزومی ندارد بدبینانه به پروژه نگاه کنید، بلکه عاقلانه و دقیق به آن بنگرید.

پیاده‌سازی

پس از اینکه پروسهٔ تحلیل مسائل تجاری را پشت سر گذاشتید، اطلاعات کافی را برای تصمیم‌گیری در زمینهٔ انجام یا عدم انجام پروژه خواهید داشت. اگر تصمیم به آغاز پروژه گرفتید، به فاز پیاده‌سازی می‌رسید. در این فاز باید طرح‌ها را عملی کرد.

تمیز کردن کد برنامه

پیاده سازی با تمیز کردن کد آغاز می‌شود تا کدی حاصل شود که قابل عرضه به عموم باشد. موارد زیر باید در تمیز کردن کد در نظر گرفته شوند:

- **آنچه که عرضه می‌کنید به سادگی قابل کامپایل باشد و تمامی کارآیی‌ها در آن گنجانده شده باشد**- جامعه متن‌باز نسبت به نرم‌افزارهایی که ناکارا باشند، واکنش منفی نشان می‌دهد. این بدان معنا نیست که نرم‌افزار باید خالی از اشکال باشد، بلکه باید چنان باشد که کاربران نحوهٔ عملکرد آن را متوجه شوند.

- **کد دارای ساختار و توضیحات-** اگر این کد را از یک پروژه بزرگتر استخراج می‌کنید، ایجاد ساختار در آن و اضافه کردن توضیحات به آن کار بسیار پیچیده‌ای خواهد بود. اما فراموش نکنید که برنامه‌نویسان خارج سازمان باید قادر به درک نحوهٔ عملکرد کد برنامه باشند. هر چه درک کد شما آسانتر باشد، علاقمندی به کار بر روی آن بیشتر می‌شود.
- **تصفیه کد-** باید مطمئن شوید که کد شما عاری از هرگونه اطلاعات خصوصی است. این اطلاعات معمولاً به صورت نام کارمندان، اسم رمز محصولات و محصولات آینده است. در میان توضیحات برنامه، ممکن است عبارت‌های موهن به کار رفته باشد که برای حفظ آبروی خود باید آنها را با دقت بیابید و حذف کنید.
- **وضعیت مالکیت معنوی کد را روشن کنید-** شما باید از اینکه کد پروژه تماماً داخل سازمان نوشته شده است و مجوز انتشار آن را دارید، مطمئن شوید. در صورتی که کدی توسط پیمانکاران شما نوشته شده باشد، باید مجوز انتشار آن را از آنها دریافت کرده باشید تا بتوانید آن کد را به سایر کدهایی که متن‌باز هستند، اضافه کنید. در این مرحله باید با یک حقوقدان باتجربه در زمینه مالکیت معنوی مشورت کنید.

از موارد دیگری که باید به همراه کد ارائه کنید، راهنمای تفضیلی کامپایل کردن کد و تشریح هرگونه وابستگی برنامه به کتابخانه‌ها می‌باشد. اگر می‌خواهید جامعه‌ای فعال در زمینهٔ پروژه شما ایجاد شود، باید کد عرضه شده از کیفیت بالایی برخوردار باشد. شما و مهندسین شما باید این مسئله را مدنظر داشته باشید.

انتخاب مجوز

این مرحله را باید با کمک مشاور حقوقی خود پیش ببرید. در فصل ۳ اشاره شد که بیش از ۳۰ مجوز مورد قبول OSI وجود دارد. باید یکی از این مجوزهای موجود را برای پروژه انتخاب کنید. اولین قدم این است که در زمینهٔ دو طرفه بودن مجوز (بدین معنا که هر کاری که روی پروژه انجام

می‌شود، باید به جامعه عرضه شود) تصمیم‌گیری کنید. اگر قصد دارید جامعه‌ای ایجاد کنید که انگیزه برای نوآوری و رشد پروژه داشته باشد، باید از این گونه مجوزها و مجوزهای دوطرفه استفاده کنید که رایج‌ترین آنها GPL و LGPL است. اما اگر می‌خواهید این امکان را در اختیار دیگران بگذارید که از نرم‌افزار شما برای تولید محصولات اختصاصی استفاده کنند، باید از مجوزهای BSD یا MIT استفاده کنید. البته همان گونه که قبلاً اشاره کردم، گزینهٔ مجوز دوگانه را نیز همچنان در اختیار دارید.

صرفنظر از نوع مجوزی که انتخاب می‌کنید، باید برای تمامی پروژه‌های متن‌باز خود، یک خط مشی ثابت تعیین کنید. منظور از خط مشی ثابت این نیست که همه پروژه‌ها تحت یک مجوز عرضه شوند، بلکه باید برای پروژه‌های از دسته مجوزهای مشابهی استفاده شود. بدین ترتیب افراد خارج از سازمان شما می‌دانند چه انتظاراتی باید داشته باشند. این خط مشی باید بخشی از سیاست کلی سازمان شود.

محیط برنامه‌نویسی

اکنون باید محیط برنامه‌نویسی، ابزار و فرآیندهای کار با جامعه را معین کنید. این مرحله ممکن است به سادگی قرار دادن پروژه در سایت و سرنزدن به آن باشد. اما این راه خوبی برای ایجاد یک جامعهٔ برنامه‌نویسان پویا نیست. اکثر برنامه‌نویسان از ابزار رایج و استفاده شده در سایر پروژه‌های متن‌باز استفاده خواهد کرد. تعدادی از این ابزار در فصلهای قبل توضیح داده شده‌اند. اما می‌توان از ابزار ساده دیگری نیز برای به اشتراک گذاشتن اطلاعات استفاده کرد، مانند ایمیل. بستگی به نوع پروژه، می‌توانید از گروه‌های خبری نیز برای تسهیل ارتباط بین برنامه‌نویسان استفاده کنید.

همچنین دربارهٔ محل نگهداری پروژه نیز باید تصمیم‌گیری کنید. می‌توانید آن را روی کارگزارهای خودتان قرار دهید یا از های میزبانی (Hosting) شرکت‌های ثالث استفاده کنید. در زمینهٔ میزان ترافیک این کارگزارها، باید به برنامه‌های بازاریابی خود مراجعه کنید.

آموزش

اغلب آموزش را جزو مراحل متن‌باز کردن نرم‌افزار در نظر نمی‌گیرند و این اشتباه است. آموزش باید در برنامه‌های بازاریابی شما در نظر گرفته شده باشد. اکثر برنامه‌نویسان قادر به شرکت در کلاس‌های آموزشی یا مسافرت به شهر دیگری برای آموزش نیستند. در صورتی که قصد تشکیل یک جامعه‌ای متشکل از مشتریان خود دارید و آموزش محصول را به آنها وعده داده‌اید، باید از طریق آموزش رسمی آنها را تعلیم دهید.

البته امکان آموزشهای الکترونیکی، از قرار دادن مطالب آموزشی در سایت گرفته تا سمینارهای گسترده اینترنتی نیز در اختیار شماست.

بازاریابی

برای کسانی که تصور می‌کنند متن‌باز کردن نرم‌افزار کار ساده‌ایست، بازاریابی نرم‌افزار نیز مسخره به نظر می‌رسد. اما پروژه‌های موفق آنهایی هستند که برنامه‌های مستند و حساب شده‌ای برای بازاریابی داشته باشند. در اینجا بخشهای اصلی برنامهٔ بازاریابی را بررسی می‌کنیم:

برنامه‌ریزی برای آغاز پروژه

از مهمترین بخشهای برنامهٔ بازاریابی، برنامه‌ریزی برای آغاز یک پروژه است. در این مرحله باید با اعضای کلیدی جامعهٔ برنامه‌نویسان مشورت کنید. اگر بتوانید از این اعضای کلیدی در اعلان عمومی برای آغاز پروژه نام ببرید به آغاز پروژه بسیار کمک خواهد کرد. اگر شرکای مشهوری هم دارید باید از آنها برای آغاز پروژه استفاده کنید. در مورد پروژه‌هایی که مربوط به یک صنعت خاص می‌شوند، باید از فرصت‌های پیش‌آمده در سمینارها و وقایع این صنعت برای معرفی پروژه خود استفاده کنید.

آگاهی جامعهٔ برنامه‌نویسان

باید آگاهی جامعه برنامه‌نویسان را در زمینهٔ پروژه بالا ببرید تا انگیزه شرکت در پروژه در آنها تقویت شود. عبارت «جامعه برنامه‌نویسان» یا به طور اختصار جامعه، یک عبارت کلی است که به هر

گروهی از برنامه‌نویسان اطلاق می‌شود. مسلماً اگر شما قصد دارید برنامه‌ای برای مدیریت منابع انسانی تولید کنید، مسلماً نیازی به هَکِرها ندارید.

همکاری رقابتی

علت اصلی کار با رقبا این است که یک شرکت به تنهایی قادر نیست هزینه‌های یک پروژه را پرداخت کند. بخصوص وقتی که پروژه به گونه‌ای است که کل جامعه از نتایج آن بهره می‌برند. در برخی صنایع کار با رقبا امری عادی و رایج است، ولی در برخی دیگر از صنایع پیوستن به رقبا در اجرای یک پروژه عجیب به نظر می‌رسد. اما اگر طبیعت باز بودن پروژه را درست درک کرده باشید و متوجه شده باشید که هر کسی در این گونه همکاری به یک اندازه سود می‌برد، مسئله شراکت با رقبا را بهتر قبول خواهید کرد. البته همچنان باید با وکیل خود برای پرهیز از رفتارهای ضدانحصار (antitrust)، مشورت کنید. زیرا قانون، نوع همکاری با رقبا را محدود کرده است.

نگهداری

فاز نگهداری با عمومی شدن پروژه آغاز می‌شود. اکنون شما برنامه‌های بازاریابی خود را با موفقیت اجرا کرده‌اید و یک جامعه‌ای از برنامه‌نویسان حول پروژه شما در حال شکل‌گیری است. این فاز را باید به دقت مدیریت کرد تا موفقیت پروژه تضمین شود. در مورد اکثر پروژه‌ها باید فعالیت‌های بازاریابی و آگاهی دادن به جامعه را همچنان ادامه دهیم.

سنجش

در فاز برنامه‌ریزی تجاری باید برخی از معیارهای تضمین کننده موفقیت را تعیین کرده باشیم. تعداد برنامه‌نویسان، کارهای عرضه شده و میزان تقاضا برای نرم‌افزار برخی از این معیارها هستند. هر یک از این معیارها را باید در بازهٔ زمانی مشخصی اندازه‌گیری کنید و سپس بر اساس نتیجه‌ای که از آن می‌گیرید، اقدام کنید. به عنوان مثال اگر تعداد برنامه‌نویسان کمتر از حد نیاز است. باید برنامه‌های بازاریابی خود را گسترش دهید. به طور کلی باید تمامی معیارها را به همین ترتیب زیر نظر داشته باشید و متناسب با آنها واکنش نشان دهید.

بازار در حال پیشرفت

نباید شرایطی برای خود ایجاد کنید که پس از آغاز پروژه قادر به سرمایه‌گذاری در زمینهٔ تبلیغ آن نباشید. در صنعت تحولات و وقایع نسبتاً زیادی اتفاق می‌افتد و پروژه شما می‌تواند از این وقایع استفاده کند. به عنوان مثال، فرض کنید پروژه‌ای در زمینهٔ برنامه‌های تلفن همراه در دست اجرا دارید و شرکتی محصول یا استاندارد جدیدی در زمینهٔ این صنعت ارائه می‌کند. شما می‌توانید با ارتباط دادن این اتفاق جدید به پروژه خود، برای آن تبلیغ کنید و آگاهی عمومی نسبت به آن را افزایش دهید.

برنامه‌نویسان تازه وارد مدام در حال پیوستن به صنعت هستند. اطلاعات سال قبل برای این تازه واردها تازگی دارد. به روز کردن و آموزش پیوسته برنامه‌نویسان، باعث می‌شود که جامعه همواره در زمینهٔ پروژه شما پویا باشد.

اهداف استراتژیک

یکی از جالب‌ترین نکات در زمینهٔ متن‌باز (که البته اکثر مدیران را نگران می‌کند) این است که جامعهٔ برنامه‌نویسان پروژه شما را به جاهایی می‌برد که تصور نمی‌کردید. قسمتهایی از نرم‌افزار شما ممکن است در جاهایی استفاده شود که پیش‌بینی آن را نمی‌کردید. در اغلب موارد به این وقایع باید از دید مثبت نگاه کنیم. اینها علامت فعالیت و رشد جامعهٔ برنامه‌نویسان است. در عین حال باید مراقب انحراف پروژه از اهداف پیش‌بینی شده آن باشیم.

در اینجا پویایی جالبی صورت می‌گیرد که نیازمند نظارت دقیق شماست. این امکان وجود دارد که جامعهٔ متن‌باز بخواهد پروژه را در جهتی غیر از آنچه که شما در نظر گرفته‌اید هدایت کند. این جهت‌گیری ممکن است بهتر از آنچه باشد که برای آن در نظر گرفته بودید. در اینجا شما باید پیشنهادها را با ذهنی باز دریافت کنید و آنها را دقیقاً بررسی کنید. به هر حال این وظیفهٔ شماست که خود را به عنوان راهبر پروژه بشناسانید و در زمینهٔ اهداف آیندهٔ پروژه مذاکره کنید. در کل باید

غرور را کنار گذارید تا فرصت‌هایی که از طریق پیشنهادهای ارائه شده، در اختیار شما قرار می‌گیرد از دست ندهید.

متن‌باز داخلی

به فرآیند به کارگیری نرم‌افزارهای موجود متن‌باز در محصولات یا پروژه‌های داخل سازمانی، «متن‌باز داخلی» گویند. قواعد و فرآیندهای مرتبط با متن‌باز داخلی با متن‌باز خارجی بسیار متفاوت است. در این فرآیند کسب و کار شما باید قوی باشد. در ادامه به برخی از انگیزه‌های متن‌باز داخلی اشاره می‌کنیم:

- **استانداردها-** اگر گونهٔ متن‌بازی از استانداردهای مورد نیاز کسب و کار شما وجود داشته باشد، بهتر است محصول خود را بر اساس آن استاندارد تهیه کنید. مزیت این کار این است که لزومی ندارد خودتان مجدداً آن استاندارد را تهیه و سپس استفاده کنید.
- **فناوری‌های موجود-** اگر جامعهٔ متن‌باز فناوری تهیه کرده باشد که صنعت آن را پذیرفته است، می‌توان با استفاده از آن فناوری سیر پیشرفت محصول خود را سریعتر کنید.
- **منابع-** اگر از فناوری‌های موجود در زیربنای محصول خود استفاده کنید، می‌توانید نیروی کار خود را بر بخشهایی متمرکز کنید که درآمدزائی بیشتری دارد.
- **خطر دوطرفه بودن(بازگرداندن کد)-** مجوزهای GPL، LGPL و تعدادی دیگر از مجوزهای متن‌باز شما را ملزم به ارائه تغییرات و اصلاحات برنامه به جامعهٔ متن‌باز می‌کند. بنابراین باید در زمینهٔ استفاده از این گونه کدها در محصول خود با دقت بیشتری تصمیم‌گیری نمایید.

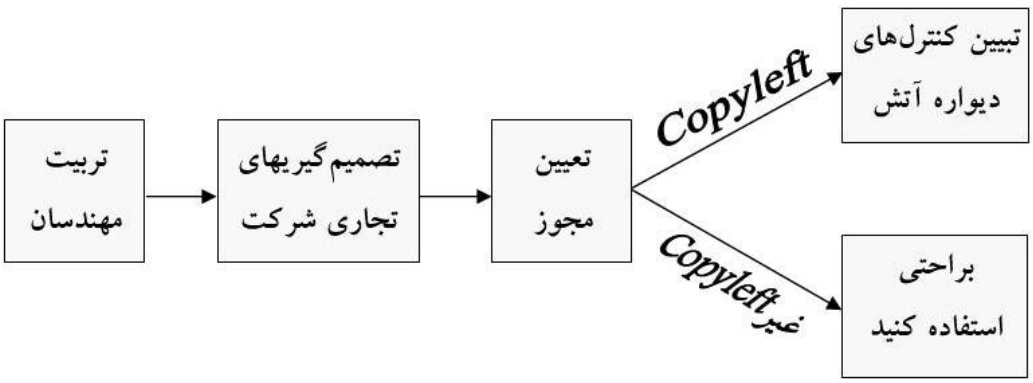
به همین ترتیب شرایطی هم وجود دارد که استفاده از متن‌باز داخلی باید پرهیز شود.

- **جهت‌گیری استراتژیک-** گرچه پروژه‌های متن‌باز می‌توانند نیازهای فوری شما را برآورده کند، اما ممکن است با اهداف آیندهٔ شما سازگاری نداشته باشد. شما باید در زمینهٔ اهداف آیندهٔ نگهدارندهٔ پروژه متن‌باز مورد نظر مذاکره کنید و بدانید که پروژه وی در آینده

به چه سمتی خواهد رفت. اگر اهداف آیندهٔ پروژه را نادیده بگیرید ممکن است مجبور شوید خودتان به تنهایی نگهداری و پشتیبانی پروژه را به عهده گیرید.

- **نیروی فنی**- اگر راهبران کلیدی پروژه مخالف استفاده از متن‌باز باشند، حداقل کاری که باید انجام شود این است که طراحی آن بازنگری شود. در چنین شرایطی باید بین نیاز به فناوری جدید و مخالفت‌های موجود تعادل برقرار کرد. در نهایت اول باید مطمئن شوید که تیم شما در جهت اهداف درآمدزا حرکت می‌کند و به تدریج گزینه‌های مربوط به متن‌باز را، با سیاست، به آنها تحمیل کنید.
- **کنترل عرضهٔ محصول**- همان گونه که می‌دانید جامعهٔ متن‌باز وقتی نرم‌افزاری را عرضه می‌کند که نگهدارندهٔ آن به این تشخیص برسد که زمان آن فرا رسیده است. اگر نیازمند کنترل دقیقی در زمان عرضهٔ محصول هستید، ممکن است نتوانید منتظر بمانید تا جامعه اصلاحات مورد نظر شما را انجام دهد. البته در چنین شرایطی یک راه این است که با جامعه همکاری کنید و خودتان اصلاحات مورد نیاز را روی پروژه انجام دهید.

در شکل ۲-۱۲ فرآیندهای مورد نیاز برای داشتن کنترل قابل قبولی بر روی پروژه متن‌باز داخلی نمایش داده شده است.



شکل ۱۲-۲: فرایند متن‌باز داخلی

همان گونه که می‌بینید اولین قدم، داشتن گروه مهندسی است که به خوبی تربیت شده باشد.

تربیت مهندسان

برای یک مهندس رفتن به اینترنت و پیدا کردن کدهای مفید و استفاده آنها در یک پروژه بسیار آسان است. مشکل اینجاست که اگر مهندسان خود را آموزش نداده باشید، ممکن است از کدهایی استفاده کنند (مثلاً کدهای دارای مجوز GPL) که وضعیت اختصاصی بودن محصول را به مخاطره بیاندازد. عده‌ای فکر می‌کنند اگر از کدهای GPL به طور مخفیانه استفاده کنند، هیچ مشکلی برایشان پیش نخواهد آمد و اگر هم به دادگاه کشیده شوند پیروز خواهند شد. این تصور کاملاً غلط است. بلکه عواقب این کار این است که شما پس از پرداخت جریمهٔ تعیین شده، مجبور خواهید شد کدهای GPL را از محصول خود خارج کنید. به هر حال اگر مهندسان خود را آموزش دهید هزینهٔ شما بسیار کمتر خواهد بود.

در کل، باید مراقب باشید کاری نکنید که مجبور به پس گرفتن محصول خود از مشتریان شوید. این مسئله از در اختیار عموم قرار گرفتن دانش فنی سازمان مهم‌تر است. به همین دلیل شما باید مهندسان خود را به خوبی آموزش دهید تا از خطرات به کارگیری کدها از منابع نامشخص و تأثیر آن بر کسب و کار شما آگاه باشید.

تصمیم‌گیریهای مربوط به سازمان

همچنان که رقابت شما با جامعهٔ متن‌باز بیشتر می‌شود، سازمان باید سیاستهایی نیز برای همراه شدن با آن داشته باشد. مسلماً این کتاب به این منظور نوشته شده است که اطلاعات کافی در زمینهٔ مزایای استفاده از نرم‌افزارهای متن‌باز در اختیار شما قرار دهد. با این حال حتی اگر شما سیاستی را دنبال کردید که کلاً متن‌باز را داخل سازمان ممنوع می‌کند، این نکته را نباید فراموش کنید که برای مهندسان استفاده از کدهای موجود در اینترنت بسیار آسان است. به همین دلیل، باید سیاستهای شما به طور واضح و روشن به آگاهی مهندسان برسد.

اگر قصد دارید استفاده از متن‌باز را آزاد گذارید، باید سیاستهای خاصی را دنبال کنید. این سیاستها باید موارد زیر را پوشش دهند:

- چه سطحی یا دسته‌ای از پروژه‌ها می‌توانند از نرم‌افزارهای متن‌باز استفاده کنند؟ مجوز یا مجوزهای این نرم‌افزارها چه باشد؟ مهندسان هنگام استفاده از منابع شرکت یا منابع موجود در اینترنت چگونه باید عمل کنند؟ این نرم‌افزار چگونه داخل سازمان مدیریت می‌شوند؟ نحوهٔ پشتیبانی نرم‌افزار عرضه شده که حاوی کدهای متن‌باز است، چگونه باشد؟

در فصل بعد مسائل مربوط به کارمندان را بیشتر بررسی خواهیم کرد که در تصمیم‌گیری بهتر به شما کمک خواهد کرد.

تعیین نوع مجوز

همان گونه که می‌دانید دسته‌ای از مجوزها وجود دارد که به صورت دوطرفه (رفت و برگشتی) هستند. یعنی شما را ملزم به بازگرداندن تغییرات انجام شده روی نرم‌افزار می‌کند. GPL و LGPL متداول‌ترین این دسته مجوزها هستند. اگر از این گونه کدها استفاده می‌کنید، باید کنترل مهندسی بسیار دقیقی بر روی پروژه داشته باشید تا بتوانید از ترکیب کدهای متن‌باز و اختصاصی جلوگیری کنید.

اگر مجوز کدهای استفاده شده به صورت دوطرفه نباشد، مانند BSD یا MIT می‌توانید به راحتی از آنها در پروژه خود استفاده کنید. اما همچنان نباید کار با جامعهٔ متن‌باز را فراموش کنید. زیرا با این کار می‌توانید آنها را در زحمت پشتیبانی نرم‌افزار شریک کنید.

دیوار آتش

اگر شرکت شما هم محصولات اختصاصی و هم محصولات متن‌باز دارد، باید دیوار آتشی بین این دو محیط ایجاد کنید. بهترین راه ایجاد این دیواره این است که از دو سیستم مدیریت نرم‌افزار جداگانه استفاده کنید یا اینکه فقط از مخزن‌های عمومی متن‌باز شناخته شده، برای این گونه برنامه‌های خود استفاده کنید. وظایف مهندسان را نیز باید به درستی تعیین کنید. اگر مهندسی بر روی هر دو نوع نرم‌افزار کار می‌کند، باید قبلاً به خوبی تربیت شده باشد تا سهواً کدهای متن‌باز را وارد کدهای اختصاصی نکند، مگر اینکه مجوز آن کد، این کار را مجاز شمرده باشد. حتی در

مواردی که تیمهای مهندسی متن‌باز و اختصاصی به طور مستقل کار می‌کنند، باز هم باید مراقب منبع کدها بود.

توسعهٔ فناوری اطلاعات

اگر قصد دارید در نرم‌افزارهای داخلی خود از متن‌باز استفاده کنید، دیگر نباید نگران مجوز کدها باشید. تمامی مجوزهای متن‌باز این کار را مجاز می‌دانند. مگر اینکه در آینده تصمیم به عرضه نرم‌افزار بگیرید. در این صورت باید مراقب عواقب منفی احتمالی این کار باشید:

- اگر کدی را استفاده کنید و تغییرات آن را باز نگردانید از آن پس باید خودتان از نرم‌افزار پشتیبانی کنید.
- اگر در هر زمان شرکت تصمیم به عرضهٔ پروژه‌های داخلی خود کند، در صورتی که این پروژه‌ها وابسته به اجزاء متن‌بازی باشد که قابل تفکیک از بخشهای اختصاصی نباشند، این کار غیرممکن خواهد بود.

بهترین راه این است که همواره با جامعهٔ متن‌باز همکاری کنید و همیشه در معماری نرم‌افزار، اجزاء متن‌باز و اختصاصی را بدرستی تفکیک کنید.

جبران زیان

اگر نرم‌افزار تجاری شما از نرم‌افزار شرکت ثالثی استفاده می‌کند، در قراردادی که با آن شرکت می‌بندید، اغلب بندهایی برای جبران زیان در نظر گرفته می‌شود. یکی از دلایل وجود این بندها این است که اگر این نرم‌افزار شرکت ثالث از مالکیت معنوی شخص دیگری بدون مجوز استفاده کرده باشد، شما متهم نشوید. طبیعت اکثر نرم‌افزارهای متن‌باز به گونه‌ای است که اجزای خود را از منابع مختلف جذب می‌کند که بسیاری از آنها قابل ردیابی نیستند.

حتی اگر یک شرکت بزرگ، نرم‌افزار متن‌بازی را به عنوان بخشی از نرم‌افزار تجاری خود به شما بفروشد، در صورت هرگونه شکایتی، آن شرکت نمی‌تواند پای شما را به میان بکشد.

اصطلاح *عدم جبران خسارت* در صنعت نرم‌افزار به تدریج در حال جا افتادن است. عدم جبران خسارت بدین معناست که به عنوان مثال اگر نرم‌افزاری را به طور رایگان از اینترنت دریافت کردید، در اغلب موارد کسی نمی‌تواند از شما شکایت کند.

جمع‌بندی

در این فصل، زیربنای فکری مورد نیاز برای تصمیم‌گیری در زمینهٔ استفاده از متن‌باز در داخل سازمان، در اختیار شما قرار گرفت. اکنون باید درک درستی از مفاهیم عرضهٔ نرم‌افزار به جامعهٔ متن‌باز و استفاده از کدهای متن‌باز در سازمان، پیدا کرده باشید. بنابراین شما تقریباً با تمامی اجزای لازم برای تعیین خط مشی‌های شرکت خود آشنا شده‌اید. آخرین بخشی از این اجزا که باید بررسی شود، منابع انسانی است. برخی از اجزا منابع انسانی به شما کمک خواهد کرد که سیاست متن‌باز شرکت خود را تکمیل کنید و سایر اجزا آن، به شما در استخدام تیم خبره و همکاری مؤثر با جامعه متن‌باز کمک می‌کند.

فصل ۱۳

منابع انسانی:

استخدام بهترین استعدادها

در بسیاری از شرکت‌ها در سیاستهای استخدامی تصریح می‌شود، هر کاری که کارمندان انجام می‌دهند متعلق به شرکت است. اما اگر کارمندی بر روی پروژه‌ای متن‌باز کار کند، تکلیف چیست؟ چگونه ممکن است کارمندی بتواند برنامهٔ متن‌باز خود را با توجه به این شرایط، به جامعهٔ متن‌باز برگرداند؟ متن‌باز نحوهٔ نگاه به منابع انسانی و ساختار سازمانی را از پایه تغییر داده است. در این فصل ما جزئیات فرآیند استخدام و نحوهٔ ارزیابی کارمندان را بررسی خواهیم کرد. همچنین به برخی از موارد لازم برای تأمین مالکیت معنوی در فرآیند استخدام خواهیم پرداخت. همچنین خواهیم

دید چگونه می‌توان جامعهٔ متن‌باز را جزئی از ساختار سازمانی تلقی کرد. در این فصل نحوهٔ مدیریت دیواره‌های آتش را نیز بررسی خواهیم کرد. هدف این فصل این است که نحوهٔ انجام موارد زیر را بیاموزیم:

- تکامل فرآیند استخدام
- مدیریت مالکیت معنوی کارمندان، شرکت و جامعهٔ متن‌باز
- همکاری با رهبران جامعهٔ متن‌باز

استخدام و کار با برنامه‌نویسان جامعهٔ متن‌باز، فرآیند دیگری است که به شما در شناخت افرادی که استخدام می‌کنید و تعیین وظیفهٔ اولیه آنها کمک می‌کند.

قراردادهای استخدامی

در استخدام مهندسانی که قبلاً در جامعهٔ لینوکس و متن‌باز کار می‌کردند و تعصب عمیقی نسبت به آن دارند، به مسایل جدیدی برمی‌خورید. چنین اشخاصی از اینکه در شرکتی استخدام می‌شوند که بر روی پروژه‌های متن‌باز کار می‌کند، بسیار هیجان‌زده خواهند شد. با این حال ممکن است انتظاراتی هم داشته باشند از قبیل اینکه فقط بر روی پروژه‌های متن‌باز کار کنند. بنابراین لازم است شرکت سیاستهای جدیدی برای استخدام این افراد تعیین کند. در این سیاستها باید موارد زیر در نظر گرفته شود:

- **پروژه‌ها**- آیا کارمندی می‌تواند کار خود را محدود به پروژه‌های متن‌باز کند؟ برای شرکت‌های بزرگی که سابقهٔ زیادی در همکاری با جامعهٔ متن‌باز دارند، این کار اشکالی ندارد. اما اگر قصد دارید که کارمندان را در پروژه‌های اختصاصی نیز درگیر کنید، این سیاست شما باید به وضوح به اطلاع آنها برسد.
- **مالکیت حق کپی**- همان گونه که در این کتاب یاد گرفتید، حتی حق کپی کدهای متن‌باز نیز مالکیت دارد و محفوظ است. در سیاستهای شما باید مشخص شده باشد که مالکیت حق کپی کارهای انجام شده توسط مهندسان، متعلق به کیست. آیا متعلق به

شرکت است یا متعلق به کارمند یا هر دو؟ در اغلب موارد شرکت باید مالکیت حق کپی کارها را داشته باشد تا بتواند در صورت نیاز از سیاست مجوز دوگانه استفاده کند. اما باید در نظر داشت در مواردی نظیر آپاچی و سامبا مجوز آنها به گونه‌ای تعیین شده است که امکان اعطای حق کپی کارهای انجام شده بر روی آنها به شخص دیگری وجود ندارد.

- **وقت آزاد مهندسان و وقت کار**- بسیاری از مهندسان در وقت آزاد خود بر روی پروژه‌های متن‌باز دیگر کار می‌کنند. ممکن است مرز بین کار شرکت و سرگرمی کارمند مشخص باشد یا نباشد. در عین حال ممکن است کاری را که یک کارمند به صورت سرگرمی آغاز کرده است، بتوان تبدیل به پروژه شرکتی کرد. به هر حال باید نحوهٔ کار کارمندان بر روی پروژه‌های متن‌باز در وقت آزاد آنها را مشخص کنید و قوانینی برای آن وضع کنید. درنظر داشته باشید در برخی مناطق، ممکن است قوانین محلی مانع محدود کردن کارمندان شود.

- **رقابت**- سیاست منابع انسانی شما احتمالاً امکان کار کارمندان با رقبا را نفی می‌کند. یکی از مزایای متن‌باز این است که می‌توان رقبا را در هزینه‌های پروژه شریک کرد (با در نظر گرفتن محدودیتهای قانون ضد انحصار). در چنین مواردی مهندسان شما می‌توانند فعالانه در پروژه‌های رقبای شما نیز کار کنند.

اگر مجموعه سیاستها و راهکارهای شما این موارد را پوشش داده باشد، شما باید بتوانید به راحتی از استعدادهای موجود در پروژه‌های متن‌باز خود استفاده کنید.

سیاستهای کار در پروژه

برای کارمندان خود اهداف کسب و کار خود را مشخص کنید. لازم است به برخی از کارمندان یادآوری کرد که کسب و کار شما باید سودآوری داشته باشد. حتی اگر بر روی پروژه‌ای کار می‌کنید که مستقیماً درآمدزا نیست، تیم آن پروژه باید بداند که چگونه کسب درآمد کند (مثلاً پشتیبانی، خدمات، سخت‌افزار و غیره). اگر مدل تجاری خود را به مهندسان توضیح دهید، آنها خواهند

توانست در حالی که به طور مؤثر با جامعهٔ متن‌باز کار می‌کنند، بین نرم‌افزارهای اختصاصی و متن‌باز مرز قائل شوند و بهتر تصمیم‌گیری کنند.

همچنین باید نحوهٔ رفتار خود با جامعهٔ متن‌باز را معین کنید. از چه مجوزهایی برای پروژه‌های خود استفاده کنید تا امکان کار کارمندان بر روی آن را بدهد؟ اگر بر روی پروژه‌های کرنل لینوکس کار می‌کنید، سیاست شما در زمینهٔ ماجول‌های اختصاصی و متن‌باز کرنل چیست؟ به طور خلاصه شما باید عناصر بررسی شده در این فصل را، کنار عناصری که در فصلهای قبل یاد گرفتید بگذارید تا بتوانید یک سیاست جامع متن‌باز برای شرکت خود تهیه کنید.

استخدام شخص مناسب

در فرآیند استخدام باید بتوانید تشخیص دهید که آیا شخص جدید مناسب سازمان شما هست یا نه؟ باید از معیارهایی که از قبل تعیین کرده‌اید به علاوه برخی معیارهای دیگر ذکر شده در این قسمت، برای تشخیص مناسب بودن یک کارمند استفاده کنید. این معیارهای اضافه را در ادامه بررسی می‌کنیم.

فناوری

هر فناوری (کرنل لینوکس، فایل سیستم، کارگزار وب، درایورها، پشته‌های شبکه و غیره) یک جامعهٔ برنامه نویس فرعی خاص خود دارد. کارمندی که استخدام می‌کنید، گرچه ممکن است در جامعهٔ بزرگ متن‌باز شناخته شده نباشد، اما ممکن است در جامعهٔ فرعی تخصصی خود مشهور باشد. بنابراین دنبال شهرت وی در جامعهٔ لینوکس و متن‌باز نباشید، بلکه به سراغ جامعه‌های فرعی بروید و میزان شهرت او را در آنجا جویا شوید. برای این کار کافیست با اعضای جامعهٔ فرعی مورد نظر تماس بگیرید و نام افراد کلیدی آن جامعه را بپرسید.

خانهٔ جامعه

هر پروژه‌ای یک نگهدارنده دارد. وظیفهٔ نگهدارنده این است که کارهای عرضه شده را قبول یا رد کند و در زمینهٔ زمان مناسب برای عرضه نرم‌افزار، تصمیم‌گیری کند. گاهی جوامع جدید در نتیجهٔ عدم توافق بین یک برنامه‌نویس و نگهدارندهٔ نرم‌افزار، شکل می‌گیرد. به این اتفاق «منشعب شدن[1]» گویند که البته به ندرت اتفاق می‌افتد. این از خوشایندترین فرآیندها در متن‌باز است. اما در مواردی این کار به صرفه نیست (مثلاً در زمینهٔ کرنل لینوکس). نگهدارنده‌های خوب آنها هستند که سابقهٔ طولانی در حل مشکلات، فناوری خاص دارند و تمامی انرژی خود را بر روی یکی از پروژه‌های آن فناوری متمرکز کرده‌اند. شما می‌توانید برای یک پروژه خاص یک نگهدارنده استخدام کنید. در چنین شرایطی باید تعادلی بین وظایف او نسبت به شرکت و نسبت به جامعهٔ متن‌باز برقرار کنید. آیا شرکت می‌تواند با مواردی که کارمندی تصمیم‌گیری می‌کند که به ضرر شرکت تمام می‌شود، برخورد کند؟ فکر نکنید صرف اینکه نگهدارنده را استخدام کرده‌اید، پروژه را تحت کنترل خود دارید. به یاد بیاورید که لینوس توروالدز نگهدارندهٔ کرنل لینوکس است نه شرکت ترانس متا، که در آنجا کار می‌کند.

نگهدارنده یا برنامه‌نویس

همان گونه که تاکنون متوجه شده‌اید، اکثر پروژه‌های متن‌باز با نگهدارنده و جامعه برنامه‌نویسان آنها شناخته می‌شوند. شما باید به خوبی بدانید که چه نقشی را در اینجا می‌خواهید بر عهده بگیرید. برای موفقیت باید ذخیره قوی‌ای از برنامه‌نویسان به همراه تعداد زیادی راهبر نزد خود داشته باشید. اگر بتوانید از افرادی استفاده کنید که در زمینه فناوری‌های مختلف فعالیت کرده‌اند، برد خوبی خواهید کرد. این افراد درک خوبی از نحوهٔ کار جوامع مختلف دارند و با نگهدارنده‌های پروژه‌های مختلف کار کرده‌اند. شما می‌توانید برای شناخت ارزش یک برنامه‌نویس با نگهدارندهٔ پروژه‌هایی که او در آنها دخیل بوده است، مشورت کنید.

---

[1] . Forking

اگر نیازمند یک راهبر برای پروژه‌ای هستید، می‌توانید از افرادی که تجربهٔ نگهداری پروژه را دارند استفاده کنید. برای محک زدن یک نگهدارنده می‌توانید از موارد زیر استفاده کنید:

- آیا او برنامه‌نویسان کلیدی را می‌شناسد و از آنها بخوبی استفاده می‌کند؟
- چگونه اختلاف و تداخل بین برنامه‌نویسان را حل می‌کند؟
- آیا او با پروژه‌های رقیبی که در نتیجه عدم توافق ایجاد شده‌اند، آشناست؟ آیا می‌شد این مشکلات را به نحوهٔ بهتری حل کرد؟
- آیا او قادر است برنامه‌نویسان کلیدی را جذب پروژه خود کند؟
- آیا او درگیر فرآیندهای چرخهٔ عرضهٔ نرم‌افزار بوده است؟
- چگونه تشخیص می‌دهد که زمان عرضهٔ نرم‌افزار فرا رسیده است؟
- آیا او می‌تواند کدهایی را به شما نشان دهد که بهتر از کارهای خودش هستند؟

در حقیقت، نگهدارنده‌ها افرادی هستند که می‌توانند با استفاده از مهارتهای فنی خود و تعامل مؤثر با برنامه‌نویسان، روند توسعهٔ یک پروژه را هدایت کنند.

شهرت و احترام در جامعه

کسی می‌تواند یک پروژه را در جامعه راهبری کند که فعال باشد و کارهای خوبی برای پروژه ارائه کرده باشد. کنترل پروژه بر عهدهٔ نگهدارنده است. نگهدارندهٔ پروژه نحوهٔ کار افراد و کیفیت آنها را می‌داند. آیا کارمندی را که می‌خواهید استخدام کنید در جامعهٔ متن‌باز از شهرت و احترام برخوردار است؟ برای تشخیص میزان شهرت و احترام او، می‌توانید به لیست‌های پستی و آرشیوهای موجود از پیام‌های رد و بدل شده میان برنامه‌نویسان مراجعه کنید.

تعامل اینترنتی

وقتی که از جوامعی که کارمند در حال استخدام، در آنها درگیر بوده است آگاه شدید و به این نتیجه رسیدید که او نیازهای شما را برآورده می‌کند، زمان آن می‌رسد که نحوهٔ تعامل او با دیگران را بررسی کنید. تعامل از طریق اینترنت، زیربنای مهارت روابط عمومی افراد است. این مسئله بسیار

حائز اهمیت است که او بتواند بر سایر افراد، در مناطق دیگر دنیا، تأثیر گذارد. برای فهمیدن این موضوع می‌توانید از موارد زیر استفاده کنید:

- به آرشیو ایمیل‌ها در لیست‌های پستی مراجعه کنید و میزان تعامل وی را بررسی کنید. در نحوهٔ تعامل وی دقیق شوید. آیا او با فرهنگ شرکت شما سازگار است؟ آیا می‌تواند در بهبود کارهای عرضه شده توسط کارمندان مؤثر باشد؟
- ببینید او از چه نام‌های مستعاری برای خود استفاده کرده است. آرشیوهای SlashDot (به بخش مرجع کتاب مراجعه کنید) و سایر آرشیوهای محلی را بررسی کنید. آیا او با استفاده از نام مستعار فریبه کاری می‌کند؟ آیا او احساسی عمل می‌کند یا منطقی؟
- آیا او در زمینه موضوعات مفید آغاز بحث می‌کند یا اینکه فقط به دیگران پاسخ می‌دهد؟ آیا پاسخهای او مفید هستند؟

سعی کنید میزان احترام او را از طریق تعداد افرادی که دنبال راهنمایی‌های وی هستند، تشخیص دهید. همچنین میزان پذیرش کارهای او توسط نگهدارنده‌ها را بررسی کنید. کسی که در جامعه دارای احترام باشد، کارهایش بدون هیچ بحثی پذیرفته می‌شود.

کارهای عرضه شده

از او بخواهید که لیست کاملی از کارهایی که در پروژه‌های مختلف متن‌باز انجام داده است، در اختیار شما قرار دهد. اگر او در پروژه اختصاصی شرکت دیگری کار کرده باشد، اغلب نمی‌توان از کیفیت و ارزش دقیق این کار وی مطلع شد. زیبایی متن‌باز در اینجاست که همه چیز در اختیار عموم است. شما می‌توانید به راحتی کیفیت، کمیت و توانایی فنی متقاضی استخدام را بسنجید. می‌توانید از مدیران و سایر کارمندان خود برای تحلیل کارهایی که او انجام داده است، طلب کمک کنید.

همچنین شما باید سبک کارهای او را نیز بررسی کنید. آیا او فقط مشکلات خاص را حل می‌کند یا در مباحث طراحی و معماری پروژه نیز دخیل است؟ سبک کارهای او باید با نیاز شما

همخوانی داشته باشد. به عنوان مثال اگر نیازمند شخصی برای پشتیبانی هستید، شخصی که در زمینهٔ حل مشکلات کار کرده است، مناسب این موقعیت می‌باشد. اما اگر به دنبال یک نگهدارنده و راهبر هستید، باید کسی را پیدا کنید که سابقهٔ درخشانی در برنامه‌نویسی و مدیریت کارهای سنگین داشته است.

جغرافیا

هر کس که در زمینهٔ کار با جامعهٔ متن‌باز تجربه داشته باشد، قادر است با سایر افراد از مناطق و فرهنگ‌های مختلف دنیا تعامل کند. وقتی که قصد استخدام شخصی را دارید، باید تعیین کنید که متقاضیان از چه مناطقی می‌توانند باشند و پس از استخدام در کجا ساکن خواهند شد.

اینکه بتوانید افراد مورد نیاز را از هر جای دنیا استخدام کنید، مزیت بزرگی برای شرکت شماست. اما داشتن تیمی که از نظر مکانی در کنار یکدیگر کار می‌کنند نیز دارای مزایای زیادی است. بر اساس تجربهٔ من، یکی از بهترین روشها این است که تیم هر پروژه، در یک نقطه متمرکز باشد. از پخش کردن پروژه‌ها در مناطق مختلف پرهیز کنید. شما می‌توانید بعدها، در هر منطقه تیم ماهری برای بومی‌سازی پروژه و پشتیبانی آن تعیین کنید. مزیت دیگر این روش این است که تیم پروژه می‌تواند با برنامه‌نویسان محلی همانند افراد جامعهٔ متن‌باز ارتباط برقرار کند.

اگر شما بر روی پروژه متن‌بازی کار می‌کنید که برای شرکت بسیار حائز اهمیت است، سعی کنید جلسات دوره‌ای با افراد کلیدی جامعهٔ متن‌باز داشته باشید. به این گونه جلسات معمولاً «جشن هک»[1] می‌گویند. اگر بتوانید این جلسه را نزدیک یکی از وقایع (سمینارها، کنفرانس‌ها) مربوط به نرم‌افزار یا کسب و کار خود برگزار کنید، می‌توانید جمع کثیری از برنامه‌نویسان را یکجا جمع کنید تا با کسب و کار و پروژه شما بیشتر آشنا شوند. سعی کنید تمامی هزینه‌های آنها را، از قبیل بلیط هواپیما، هتل و غذا، بپردازید. شما متوجه خواهید شد که شرکت‌کنندگان در این جلسات بسیار قانع هستند و بیشتر دوست دارند برنامه‌نویسی کنند تا اینکه پول شما را به هدر

[1] . Hack-Fest

دهند. هدف شما باید این باشد که هر آنچه که برنامه‌نویسان لازم دارند در اختیار آنها قرار دهید و یک چرخهٔ تولید جدید را آغاز کنید. به این ترتیب از همکاری این افراد نتیجهٔ بسیار خوبی خواهید گرفت.

مراحل را بشمارید

یک جامعه عمدتاً از طریق انگیزه‌هایش حرکت می‌کند. مهمترین کسی که می‌تواند در افراد پروژه انگیزه ایجاد کند، نگهدارندهٔ آن است. در زمینه پروژه‌های بزرگ سلسله مراتبی از اعتماد به طور غیررسمی بین برنامه‌نویسان و نگهدارنده وجود دارد. مسلماً نباید انتظار داشت که همیشه براحتی می‌توان نگهدارنده یک پروژه دیگر و برنامه نویسان او را به استخدام خود درآورد. در جریان استخدام یک برنامه‌نویس، باید این سلسله مراتب را کشف کنید و تعداد برنامه‌نویسان بین او و نگهدارنده را بشمارید. به این کار «شمردن مراحل»[1] می‌گویند. همان گونه که در مسیردهی ترافیک شبکه انجام می‌شود. هر چه مراحل کمتر باشد، متقاضی استخدام به نگهدارنده نزدیک‌تر است.

ساختاربندی تیمها

ساختاربندی تیمها بستگی به نقش شرکت شما در یک پروژه متن‌باز دارد. همان گونه که در مفهوم بازار شرکتی یاد گرفتید، شما می‌توانید یک پروژه متن‌باز بسیار بزرگ را داخل شرکت ساختاربندی کنید و سپس تیم حاصله را مدیریت کنید. اما اغلب شما بر روی پروژه‌های کوچک کار می‌کنید و معمولاً شما یکی از ارائه کنندگان کد به پروژه هستید.

بخاطر مسألهٔ دوطرفه بودن (بازگرداندن کد) که در فصل قبلی به آن پرداخته شد، شما باید تمام تلاش خود را بکنید تا برنامه‌نویسان برای پروژه‌های متن‌باز خارج سازمان نیز کار کنند.

مورد خاص جوامع دروازه‌ای نیز در ساختاربندی تیمها نقش دارد. ممکن است شما درگیر ترکیبی از پروژه‌های مختلف از انواع اختصاصی، متن‌باز و پروژه‌های جوامع دروازه‌ای باشید. این

---

[1] . Counting the Hops

وظیفه شماست که فرآیندهای خاصی را برای روشن کردن مالکیت معنوی کارها و مرز بین آنها تعبیه کرده باشید.

استخدام راهبران برجسته

در اینجا من یکی از تجربه‌های شخصی خود را در زمینهٔ HP بازگو می‌کنم. به عقیده بسیاری، استخدام بروس پرنز، یکی از برجسته‌ترین رهبران جامعهٔ متن‌باز، کار خطرناکی بود که من انجام دادم. اولویت اول بروس این بود که نمایندهٔ جامعهٔ متن‌باز باشد و جنبش متن‌باز را ترقی دهد. او در حال حاضر با تعدادی از تیمهای داخلی HP کار می‌کند و آنها را در زمینهٔ کار با جامعهٔ متن‌باز آموزش می دهد. گرچه بروس HP را تبلیغ و معرفی می‌کند، در موارد بسیار نادری، نظرات او با نظرات شرکت متفاوت است. من و بروس چند قانون برای رفع چنین مواردی وضع کرده‌ایم:

- بروس حق ندارد به HP و مدیران آن حمله کند.
- بروس حق ندارد اطلاعات محصولات اختصاصی را در اختیار عموم قرار دهد.
- بروس باید مشخص کند که چه موقع نظر شخصی خود را اعلام می‌کند و چه موقع به نمایندگی از HP صحبت می‌کند.

این وظیفه من بود که با صنعت، مطبوعات، تحلیل‌گران، مشتریان، شرکا و رقبا کار کنم و از اینکه آنها از نقش بروس در HP آگاهند مطمئن شوم. به نظر من گرچه نقش دوگانهٔ بروس همچنان مشکلاتی را برای برخی از تیمهای مدیریتی ایجاد می‌کند، اما برای کسب و کار HP در زمینه متن‌باز و لینوکس بسیار مفید بوده است. بسیاری از افراد در این صنعت توانسته‌اند به خوبی از نقش دوگانهٔ راهبران بهره گیرند. بروس و HP توانسته‌اند اعتبار خوبی برای یکدیگر کسب کنند.

راهبران متن‌باز اول باید جامعهٔ خود را معرفی کنند، سپس شرکت را. در ابتدا پذیرفتن چنین اصلی برای شما مشکل است، اما اگر مرزهای روشنی برای نحوهٔ ارتباطات مشخص کنید، موفقیت‌آمیز خواهد بود. کارمندان شما باید نقش منحصر به فرد چنین اشخاصی را درک کنند و

بدانند قوانینی که برای این اشخاص برقرار است، برای سایر کارمندان صدق نمی‌کند. سعی کنید رابطه‌ای مبتنی بر اعتماد با تمامی راهبرانی که قصد استخدام آنها را دارید برقرار کنید.

جمع‌بندی

ساختاربندی، شکل‌دهی و مدیریت تیمهای مهندسی که پروژه‌های متن‌باز و غیرمتن‌باز را ترکیب می‌کند بسیار متفاوت است. از نظر فرهنگی مهندسان شما بین وفاداری به جامعه یا شرکت به مشکل برمی‌خورند. شما باید از مشکل وفاداری دوگانه میان برنامه‌نویسان خود آگاه باشید. کارمندان به تدریج با آن کنار می‌آیند و دیگر هراسی از آن نخواهند داشت. در این فصل با معیارهای جدیدی برای استخدام افراد آشنا شدید. همچنین مواردی که باید در سیاست‌گذاریهای برنامه‌نویسی شرکت در نظر گرفته شود، مورد بحث قرار گرفت.

کالایی=Commodity

ساماندهی=Arrange

انجمن (وقتی که اشاره به یک گروه خاص باشد)=community

جامعه متن‌باز (وقتی که اشاره به کل جامعه متن‌باز باشد)=community

نمایش پنجره=windowing

توکار – نهفته – **درونه‌ای**=embedded

نسخه‌برداری=copy

دریافت از اینترنت=download

اعمال تغییرات بر حسب خواسته و نیاز=customization

راه حل، رویکرد، نرم‌افزار...=solution

در تله افتادن=Lock-in