پیشگفتار

ماجراجویی من برای یادگیری و درک بهتر لینوکس در سال ۱۹۹۸ آغاز شد. من به تازگی اولین توزیع لینوکس خودم رو نصب کرده بودم و به سرعت مجذوب کل مفهوم و فلسفه پشت لینوکس شده بودم. همیشه راه های بسیاری برای انجام یک کار وجود داره، میتونیم این را در مورد لینوکس هم بگوییم. توزیع های فراوانی در طول سالیان وجود داشتند که برخی از آنها هنوز هم موجوده و برخی هم به توزیع های دیگه ای تبدیل شدند[۱]. همه آنها کارها را به گونه ای متفاوت انجام می دهند تا نیازهای مخاطبان خودشون رو برآورده کنند. چون که راه های زیادی برای رسیدن به هدف یکسان وجود داره، کم کم متوجه شدم که دیگری نیازی به محدود شدن به یک پیاده‌سازی خاص از لینوکس[۲] ندارم. قبل از آشنایی با لینوکس، اصلاح مشکلات در سیستم عامل های دیگه معمولاً دشوار بود، چون انتخاب دیگه ای نداشتید. چه خوشتون بیاد یا نه. با لینوکس، مفهوم "انتخاب" شروع به ظهور کرد. اگر چیزی را دوست نداشتید، شما آزاد بودید، حتی تشویق می‌شدید که آن را تغییر دهید. من چندین توزیع مختلف لینوکس رو امتحان کردم، اما نتونستم یکی از اونها رو انتخاب کنم. هر کدوم از این سیستم‌ها به نوبه خودشون عالی بودن. دیگه مسئله درست یا غلط بودن، نبود. به یک سلیقه شخصی تبدیل شده بود. با تمام این گزینه های موجود، برام روشن شد که هیچ سیستمی نخواهد بود که برای من کاملاً ایده‌آل باشه. پس تصمیم گرفتم که سیستم لینوکس خودم رو بسازم که کاملاً با سلیقه‌های شخصی من مطابقت داشته باشه برای اینکه کاملا سیستم رو شخصی سازی کنم، تصمیم گرفتم همه چیز رو از سورس کد به جای استفاده از پکیج های باینری آماده[۳] کامپایل کنم. این سیستمه "کامل" لینوکسی

---

۱. مثل لینوکس Mint که اولین بار درسال ۲۰۰۶ برپایه Kubuntu منتشر شد ولی سال ۲۰۱۰ نسخه دبیان ادیشن اون رو منتشر کردن شد.

۲. لینوکس به صورت متن‌باز است و افراد یا سازمان‌های مختلف می‌توانند نسخه‌های سفارشی خود را بر اساس نیازهای خاص خود بسازند. بنابراین، "پیاده‌سازی خاص لینوکس" در اینجا به معنای نسخه‌های مختلف لینوکس است که توسط افراد یا جوامع مختلف توسعه داده شده‌اند.

۳. پکیج های باینری از قبل کامپایل شده به فایل های آماده اجرا(executable) میگن که توسط کامپایل شدن سورس کد یک برنامه، به ماشین کد ایجاد شده.

قدرت‌های مختلفی از انواع مختلف سیستم‌ها رو بدون داشتن نقاط ضعف مشهود اونها خواهد داشت. اولش، این ایده خیلی سخت به نظر می‌رسید. اما من به ایده اینکه چنین سیستمی قابل ساختن است، پایبند ماندم. پس از برطرف کردن مشکلاتی مانند وابستگی‌های دایره‌ای[۴] و خطاهای زمان کامپایل، سیستم لینوکس سفارشی‌سازی شده‌ای را ساختم. این سیستم کاملاً عملی و بدون مشکل مانند هر یک از سیستم‌های دیگر لینوکس در آن زمان بود. اما این سیستم ساخت دست خودم بود. احساس رضایت بخشی بود که بتونم چنین سیستمی رو خودم سرهم کنم. تنها چیزی که میتونست این رو بهتر کنه این بود که نرم افزارش رو هم خودم مینوشتم. با اشتراک‌گذاری اهداف و تجربیاتم با سایر اعضای جامعه لینوکس، فهمیدم که علاقه‌ای پایدار به این جور ایده‌ها وجود دارد. به سرعت مشخص شد که سیستم‌های لینوکسِ سفارشی‌سازی شده نه تنها برای تأمین نیازهای خاص کاربران، بلکه به عنوان یک فرصت آموزشی مناسب برای برنامه‌نویسان و مدیران سیستم به منظور بهبود مهارت‌های لینوکس (موجود) نیز عمل می‌کنند. از این علاقه گسترده، پروژه Linux From Scratch به وجود آمد. این کتاب، محور اصلیه پروژه Linux From Scratch است. این کتاب الگو و دستورالعمل‌های لازم برای طراحی و ساخت سیستم خودتون را فراهم می‌کند. اگرچه این کتاب یک الگویی را ارائه می‌دهد که درنهایت به یک سیستم کاری صحیح منجر می‌شود، اما شما آزاد هستید تا دستورالعمل‌ها را به دلخواه خودتون تغییر بدید که این نیز بخشی از این پروژه مهم است. همه چیز تحت کنترل شماست؛ ما فقط به شما کمک می‌کنیم تا سفر خودتون را آغاز کنید. امیدوارم که وقت خوبی را در کار با سیستم خود داشته باشید و از مزایای بی‌شمار داشتن یک سیستم که کاملاً برای خودتان است لذت ببرید.

مخاطب

دلایل مختلفی وجود دارد که چرا میخواهید این کتاب را بخوانید. یکی از سوالاتی که بسیاری از افراد مطرح میکنند این است : چرا باید تمام دردسرهای ساخت یک سیستم لینوکس را از اول تجربه کنیم، در حالی که میتوانیم یک سیستم موجود را دانلود و نصب کنیم؟ یکی از دلایل مهم وجود این پروژه، کمک به یادگیری نحوه عملکرد یک

---

۴. وابستگی های دایره ای (Circular dependency) به مجموعه ای از ماژول هایی میگن که برای کارکرد صحیح به همدیگه به طور مستقیم یا غیرمستقیم نیاز دارند.

سیستم لینوکس از درون به بیرون است. ساخت یک سیستم LFS نشان میدهد که چه چیزی لینوکس را به کار میاندازد و چگونه اجزای آن با یکدیگر همکاری میکنند و به یکدیگر وابسته هستند. یکی از بهترین چیزهایی که این تجربه یادگیری میتواند ارائه دهد، قابلیت سفارشی سازی یک سیستم لینوکس به منظور تناسب با نیازهای منحصر به فرد خودتان است. یکی از مزایای کلیدیLinux From Scratch این است که به شما کنترل کاملی از سیستم را بدون وابستگی به پیاده‌سازی لینوکس دیگران می‌دهد. با LFS ، شما در مقام راننده قرار دارید. شما هر جزئی از سیستم خود را تعیین می‌کنید. LFS به شما امکان ایجاد سیستم‌های لینوکس بسیار کم حجم را می‌دهد. در توزیع‌های دیگر، شما اغلب مجبور به نصب بسیاری از برنامه‌ها هستید که نه تنها از آن‌ها استفاده ای نمی‌کنید، بلکه کاربرد بسیاری از آنها را نیز نمیدانید. این برنامه‌ها منابع سیستم شمارا را تلف می‌کنند. ممکن است بگویید که با هارددرایوها و پردازنده‌های امروزی، دیگر نگران هدررفتن منابع نیستیم. ولی گاهی اوقات، باز هم محدودیت‌هایی به خاطر اندازه سیستم داریم، حداقلش همینه. به سی‌دی‌های بوت‌شدنی، فلش درایوها، و سیستم‌های تعبیه‌شده فکر کنید. اینجاهاست که سیستم (LFS) می‌تونه مفید باشه. یکی از مزایای سیستم‌های لینوکس سفارشی سازی شده، امنیت است. با کامپایل کردن کل سیستم از سورس کد، شما قادر به بررسی همه چیز و اعمال تمامی پچ‌های امنیتی که می‌خواهید خواهید بود. شما نیازی ندارید که منتظر شخص دیگری باشید تا پکیج های باینری را کامپایل کند تا یک آسیب امنیتی(Security hole) را برطرف کند. مگر اینکه خودتان پچ را بررسی کرده و پیاده‌سازی کنید، هیچ تضمینی وجود ندارد که پکیج باینری جدید به درستی کامپایل شده باشه و مشکل را به طور کامل رفع بکند. هدف از Linux From Scratch، ساخت یک سیستم کامل و قابل استفاده در سطح پایه ای است. اگر نمی‌خواهید سیستم لینوکس خود را از ابتدا بسازید، باز هم می‌توانید از اطلاعات موجود در این کتاب بهره‌برداری کنید. دلایل زیادی برای ساخت سیستم LFS خود وجود دارد که نمی‌توان همه‌ی آن‌ها را در اینجا آورد. در نهایت، آموزش به عنوان مهم‌ترین دلیل شناخته می‌شود. همچنین، در طول تجربه‌ی LFS خود، به قدرتی که اطلاعات و دانش به همراه دارند، پی خواهید برد.

معماری‌های هدف LFS

معماری‌های اصلیه هدف LFS، پردازنده‌های (32-bit)X86 AMD/Intel و X86_ (64-bit)X86_64 هستن. از طرف دیگه، دستورالعمل‌های این کتاب هم، با تغییراتی روی پردازنده‌های Power PC و ARM هم جواب میدن. برای ساخت یک سیستم که از این پردازنده‌های جایگزین استفاده می‌کنه، شرط اصلی، علاوه بر مواردی که تو صفحه بعدی هست، داشتن یک سیستم لینوکس موجود مثل اوبونتو، رد هت/فدورا، SuSE یا هر توزیع دیگه‌ای که به اون معماری مربوط می‌شه، هست. (توجه داشته باشید که یک توزیع ۳۲بیتی می‌تونه روی یک کامپیوتر ۶۴ AMD/Intel بیتی نصب و به عنوان یک سیستم میزبان استفاده بشه.) سودی[۵] که از ساختن Lfs روی یک سیستم ۶۴-بیتی به دست میاد، در مقایسه با یک سیستم ۳۲-بیتی، خیلی کمه. مثلاً، تو یک آزمایش برای ساخت LFS نسخه ۹.۱ روی یک سیستم مبتنی بر پردازنده i7-۴۷۹۰ Core که از ۴ هسته استفاده می‌کرد، آمار زیر اندازه‌گیری شد:

معماری	زمان ساخت	اندازه ساخت
۳۲-بیت	۲۳۹.۹ دقیقه	۳.۶ گیگابایت
۶۴-بیت	۲۳۳.۲ دقیقه	۴.۴ گیگابایت

همونطور که می‌بینید، روی همون سخت‌افزار، زمان ساخت ۶۴-بیتی فقط ۳٪ سریع‌تر (و ۲۲٪ بزرگ‌تر) از ساخت ۳۲-بیتیه. اگه قصد دارید LFS رو به عنوان یک سرور LAM یا یک فایروال استفاده کنید، یک پردازنده ۳۲-بیتی شاید کافی باشه. از طرف دیگه، چندین پکیج در BLFS حالا بیشتر از ۴ گیگابایت رم نیاز دارن تا ساخته شن و/یا اجرا بشن؛ اگه قصد دارید LFS رو به عنوان یک دسکتاپ استفاده کنید، نویسندگان LFS پیشنهاد می‌کنن که یک سیستم ۶۴-بیتی بسازید. ساخت(Build) پیش‌فرض ۶۴-بیتی که از LFS به دست میاد، یک سیستم "خالص" ۶۴-بیتیه. یعنی فقط اجرای برنامه‌های

---

۵. در این متن، "سود" به مزایای استفاده از یک سیستم ۶۴-بیتی در مقایسه با یک سیستم ۳۲-بیتی اشاره دارد. به طور کلی، سیستم‌های ۶۴-بیتی می‌توانند از حافظه بیشتری استفاده کنند و داده‌ها را سریع‌تر پردازش کنند. با این حال، اگر برنامه‌ها و کاربردهایی که استفاده می‌کنید نیاز به حافظه بیشتر از ۴ گیگابایت ندارند، ممکن است تفاوت قابل توجهی در عملکرد بین سیستم‌های ۳۲-بیتی و ۶۴-بیتی مشاهده نکنید. در نتیجه، "سود" یا مزیت استفاده از سیستم ۶۴-بیتی در این شرایط خیلی کم است

۶۴-بیتی رو پشتیبانی می‌کنه. ساخت یک سیستم چند کتابخانه‌ای[۶] نیاز به کامپایل کردن بسیاری از برنامه‌ها دو بار داره، یک بار برای سیستم ۳۲-بیتی و یک بار برای سیستم ۶۴-بیتی. این به طور مستقیم در LFS پشتیبانی نمی‌شه چون با هدف آموزشی ارائه حداقل دستورالعمل‌های لازم برای یک سیستم لینوکس پایه تداخل داره. برخی از ویراستاران LFS/BLFS یک شاخه چند کتابخانه‌ای از LFS رو نگهداری می‌کنن که در آدرس اینترنتی زیر قابل دسترسی میباشد :
https://www.linuxfromscratch.org/~thomas/multilib/index.html

پیش نیازها

ساخت یک سیستم LFS کار ساده‌ای نیست. نیاز به دانش قبلی خاصی از مدیریت سیستم یونیکس داره تا بتونید مشکلات رو حل کنید و دستورات لیست شده رو به درستی اجرا کنید. حداقل ، باید قبلاً بدونید چطور از خط فرمان (شل) برای کپی یا جابجایی فایل‌ها و دایرکتوری‌ها، لیست کردن محتویات دایرکتوری و فایل‌ها، و تغییر دایرکتوری فعلی استفاده کنید. همچنین انتظار میره که بدونید چطور نرم‌افزار لینوکس رو استفاده و نصب کنید. چون کتاب LFS حداقل این سطح از مهارت رو فرض می‌کنه، انجمن‌های پشتیبانی مختلف LFS احتمالاً کمک زیادی در این زمینه‌ها به شما نخواهند کرد. خواهید دید که سوالاتتون در مورد چنین دانش پایه‌ای احتمالاً بی‌پاسخ می‌مونه یا فقط به لیست مطالعه ضروری LFS ارجاع داده می‌شوید.

---

۶. چندکتابخانه ای یا همون multilib یعنی یک سیستم عامل که می‌تونه همزمان از کتابخانه‌های ۳۲ بیتی و ۶۴ بیتی پشتیبانی کنه. این امکان رو می‌ده که یک برنامه بتونه هر دو نوع کتابخانه رو ارائه بده. مثلاً، اگه یک برنامه‌نویس بخواد یک برنامه‌ای بسازه که روی هر دو نوع سیستم ۳۲ بیتی و ۶۴ بیتی کار کنه، با استفاده از مولتی لیب می‌تونه این کار رو انجام بده. ولی مشکل اینجاست که هر سیستم عاملی به شکل متفاوتی از مولتی لیب پشتیبانی می‌کنه، پس برنامه‌نویس‌ها باید برای هر سیستم عامل، تنظیمات خاصی رو انجام بدن.

قبل از ساخت یک سیستم LFS ، ما از شما می‌خواهیم که این مقالات را بخوانید:

- راهنمای ساخت نرم‌افزار:

  https://tldp.org/HOWTO/Software-Building-HOWTO.html

  این یـــــک راهنمای جامع بـــــرای ساخت و نصب پکیج های نرم‌افزاری "عمومی" یونیکس زیر لینوکس است. اگرچه مدتی پیش نوشـــته شـــده، اما هنوز خلاصه خوبی از تکنیک‌های پایه‌ای استفاده شده برای ساخت و نصب نرم‌افزار را ارائه می‌دهد.
- راهنمای مبتدیان برای نصب از منبع:

  https://moi.vonos.net/linux/beginners-installing-from-source

  این راهنما خلاصه خوبی از مهارت‌ها و تکنیک‌های پایه‌ای مـورد نیاز بـرای ساخت نرم‌افزار از سورس کد را فراهم می‌کند.

LFS و استاندارد ها

ساختار LFS تا حد امکان از استانداردهای لینوکس پیروی میکنه. اســتانداردهای اصــلی عبارتند از:

- POSIX.1-2008
- Filesystem Hierarchy Standard (FHS) Version 3.0
- Linux Standard Base (LSB) Version 5.0 (2015)

LSB شامل چهار مشخصه جداگانه است: هسته، دسکتاپ، زبانهای اجرایی(ران تــایم)، و تصویربرداری. برخی قسمتهای مشخصههای هسته و دسکتاپ مختص معماری هســتند. همچنین دو مشخصه آزمایشــی وجـود دارد: Gtk3 و گرافیـک. LFS تلاش میکنــه تــا با مشخصههای LSB برای معماریهای IA32 (32-bit x86) بیتی یا AMD64 (x86_64) که در بخش قبلی بحث شد، مطابقت داشته باشه.

توجه: خیلی ها با این پیشنیاز ها موافق نیســتن. هـدف اصــلی LSB اینــه کــه اطمینــان حاصل کنه نرم افزارهای اختصاصی(انحصاری) میتونن روی یک سیستم مطـابق، نصب و

اجرا بشن. چون LSB بر پایه منبع (سورس) هست، کاربر کنترل کـاملی روی پکیج هایی که میخواد داره؛ شما ممکنه تصمیم بگیرید بعضی از پکیچهایی که توسط LSB مشخص شدن رو نصب نکنید. درسته که میشه یک سیستم کاملی ساخت که تستهای گواهینامه LSB رو از ابتدا پاس کنه، اما این کار بدون پکیج های اضافی که خـارج از دامنـه کتـاب LFS هستن، امکانپذیر نیست. دستورالعملهای نصب این پکیچهای اضافی رو میتونیـد در BLFS پیدا کنید.

پکیچ های ارایه شده توسط LFS برای برآورده کردن نیاز های LSB :

LSB CORE	Bash, Bc, Binutils, Coreutils, Diffutils, File, Findutils, Gawk, Grep, Gzip, M4, Man-DB, Ncurses, Procps, Psmisc, Sed, Shadow, Tar, Util-linux, Zlib
LSB Desktop	None
LSB Runtime Languages	Perl, Python
LSB Imaging	None
LSB Gtk3 and LSB Graphics (Trial Use)	None

دلیل وجود پکیج ها در این کتاب

هدف از LFS ساخت یک سیستم پایه ای و قابـل اسـتفاده اسـت کـه شـامل تمام پکیج های لازم برای تکرار خودش[۷] و فراهم کردن یـک پایه نسـبتاً حـداقلی کـه از آن میتوان یک سیستم کامل تر را بر اساس انتخاب های کاربر شخصی سازی کرد. این به این معنـا نیست که LFS کوچکـترین سیسـتم ممکن اسـت. چنـدین پکیج مهم وجـود دارد کـه بـه

۷. سیستم عامل هایی مانند LFS که قابلیت "تکرار خود(self replicate)" را دارند، این امکان را به کاربران می دهنـد کـه یـک کپی دقیق از سیستم عامل فعلی خود را بسازند. این ویژگی در موارد زیر می تواند مفید باشد:

بازیابی سیستم: اگر سیستم عامل به دلیل خطا یا مشکلی دیگر ناپدید شود، می توانید از کپی ایجاد شده برای بازیابی استفاده کنید.

انتقال سیستم: اگر می خواهید سیستم عامل خود را به یک دستگاه جدید منتقل کنید، می توانید از کپی ایجاد شده استفاده کنید.

آزمایش و توسعه: اگر می خواهید تغییرات یا به روزرسانی های جدید را بر روی سیستم عامل خود امتحان کنید، می توانید این کار را بر روی یک کپی از سیستم عامل انجام دهید، بدون اینکه نگران ایجاد مشکل در سیستم عامل اصلی خود باشید.

بیان دقیق، موردنیازمان نیستند. لیست زیر دلایل وجـود هر پکیج در کتـاب را توضـیح میدهد.

- Acl این پکیج شـامل ابزارهایی بـرای مـدیریت لیسـت‌های کنـترل دسترسـی ادمین است که برای تعریـف مجـوز دسترسـی اختیاری دقیـق بـرای فایل‌ها و دایرکتوری‌ها استفاده می‌شود.
- Attr این پکیج شـامل برنامه‌هایی بـرای مـدیریت ویژگی‌های توسـعه‌یافته روی file system objects است.
- Autoconf این پکیج برنامه‌هایی را برای تولید اسکریپت‌های شل که می‌تواننـد به طور خودکار سورس کـد را از یـک الگـوی توسـعه‌دهنده پیکربنـدی کننـد، فراهم می‌کند. اغلب برای بازسازی یک پکیج پس از به‌روزرسانی روش ساخت[۸] نیاز است.
- Automake این پکیج حاوی برنامه‌هایی برای تولید فایل‌های Make از یک الگـو است. اغلب برای بازسازی یک پکیج پس از به‌روزرسانی روش ساخت نیاز است.
- Bash این برنامـه یـه جـور دسـتورالعمل بـه کـامپیوتره کـه بهش میگن شـل. خیلی‌ها ازش استفاده می‌کنن و کارهای زیادی باهاش میشه کرد.
- Bc این یکی بـرای حساب و کتاب‌های دقیقـه، مثلاً وقـتی می‌خـوایم هسـته لینوکس(kernel) رو بسازیم، بهش نیاز داریم.
- Binutils این پکیج شامل ابزارهاییه که به ما کمک می‌کنه کدهای برنامه‌نویسی رو به برنامه‌های قابل اجرا تبدیل کنیم.
- این برنامـه یـه نـوع ابـزار بـرای ساخت برنامه‌های دیگه‌سـت، مخصوصاً بـرای برنامه‌هایی که تو LFS هستن.
- Bzip2 این برنامه برای فشرده‌سازی و باز کردن فایل‌های فشرده استفاده میشه
- Check این پکیج برای تست کردن برنامه‌های دیگه به کار میره

---

۸. "روش ساخت" یا همان build procedure به فرآیند تبدیل سورس کد به یک برنامه قابل اجـرا اشـاره دارد. این یک سری مراحل است که توسعه‌دهندگان برای کامپایل و اتصال کد دنبال می‌کنند تـا بتوانـد روی کامپیوتر اجرا شود

- Coreutils این پکیج شامل برنامه‌های ضروری برای دیدن و دستکاری فایل‌ها و پوشه‌هاست. اینا برای مدیریت فایل‌ها تو خط فرمان و نصب پکیج‌های دیگه LFS لازمن
- DejaGNU این پکیج یه چهارچوب برای تست کردن برنامه‌های دیگه فراهم می‌کنه.
- Diffutils این پکیج برنامه‌هایی داره که نشون میدن بین فایل‌ها یا پوشه‌ها چه تفاوت‌هایی هست. میشه ازشون برای درست کردن پچ استفاده کرد و تو خیلی از روش‌های ساخت پکیج‌ها به کار میرن.
- E2fsprogs این پکیج ابزارهایی برای مدیریت سیستم‌های فایل ext2، ext3 و ext4 داره که خیلی رایج و تست‌شده‌اند.
- Expat این پکیج یه کتابخانه کوچیک برای تجزیه و تحلیل فایل‌های XML داره که برای ماژول XML::Parser در زبان برنامه‌نویسی Perl لازمه. Expect این پکیج برنامه‌ای داره که می‌تونه دیالوگ‌های نوشته شده رو با برنامه‌های دیگه اجرا کنه. معمولاً برای تست کردن برنامه‌های دیگه استفاده میشه.
- File این پکیج یه ابزار داره که نوع فایل‌ها رو تشخیص میده. چندتا از پکیج‌ها برای ساختنشون به این ابزار نیاز دارن.
- Findutils این پکیج برنامه‌هایی داره که بهت کمک می‌کنن فایل‌ها رو توی کامپیوتر پیدا کنی. توی خیلی از دستورالعمل‌های ساخت پکیج‌ها استفاده میشه.
- Flex این پکیج یه ابزار داره که برنامه‌هایی می‌سازه که الگوهای متنی رو تشخیص میدن. این نسخه‌ی GNU از برنامه‌ی (lexical analyzer) هست و برای ساخت چندتا از پکیج‌های LFS لازمه.
- Gawk این پکیج برنامه‌هایی برای دستکاری فایل‌های متنی داره. این نسخه‌ی GNU از برنامه‌ی awk هست و توی دستورالعمل‌های ساخت خیلی از پکیج‌ها استفاده میشه
- GCC این مجموعه‌ی کامپایلرهای GNU هست. شامل کامپایلرهای C و C++ و چندتا دیگه که توسط LFS ساخته نمیشن

- GDBM این پکیج کتابخانه‌ی مدیریت دیتابیس GNU رو داره. توسط یکی دیگه از پکیج‌های Man-DB ،LFS، استفاده میشه
- Gettext این پکیج ابزارها و کتابخانه‌هایی برای بین‌المللی‌سازی و محلی‌سازی خیلی از پکیج‌ها فراهم می‌کنه.
- Glibc این پکیج کتابخانه‌ی اصلی C رو داره. برنامه‌های لینوکس بدون این کتابخانه کار نمی‌کنن.
- GMP این پکیج کتابخانه‌های ریاضی رو ارائه می‌ده که توابع مفیدی برای حساب دقیق ارائه می‌دهند. برای ساخت GCC لازمه
- Gperf این پکیج یه برنامه تولید می‌کنه که یه تابع هش کامل از یک مجموعه کلیدها تولید می‌کنه. برای Udev لازمه
- Grep این پکیج برنامه‌هایی برای جستجو توی فایل‌ها داره. این برنامه‌ها توی دستورالعمل‌های ساخت اکثر پکیج‌ها استفاده میشن.
- Groff این پکیج برنامه‌هایی برای پردازش و فرمت‌دهی متن داره. یکی از کارهای مهم این برنامه‌ها فرمت کردن صفحات راهنما (Man Pages) هست.
- GRUB این بوت لودر یکپارچه ی بزرگه. از بین چندین بوت لودر موجود، انعطاف‌پذیرترینه
- Gzip این پکیج برنامه‌هایی برای فشرده‌سازی و باز کردن فایل‌های فشرده داره. برای باز کردن خیلی از پکیج‌ها در LFS لازمه
- Iana-etc این پکیج داده‌هایی برای خدمات و پروتکل‌های شبکه فراهم می‌کنه. برای فعال‌سازی قابلیت‌های شبکه‌ای درست لازمه
- Inetutils این پکیج برنامه‌هایی برای مدیریت شبکه‌های ابتدایی داره. Intltool این پکیج ابزارهایی برای بیرون کشیدن رشته‌های قابل ترجمه از فایل‌های منبع فراهم می‌کنه
- IProute این پکیج برنامه‌هایی برای شبکه‌های پیشرفته IPv4 و IPv6 داره. به خاطر توانایی‌های IPv6 از پکیج‌های دیگه انتخاب شده.
- Kbd این پکیج فایل‌های جدول کلید، ابزارهای کیبورد برای کیبوردهای غیر آمریکایی و تعدادی فونت کنسول تولید می‌کنه.

- Kmod این پکیج برنامه‌هایی برای مدیریت ماژول‌های هسته لینوکس داره.
- Less این پکیج یک نمایشگر فایل متنی خیلی خوب داره که اجازه می‌ده بالا و پایین بری وقتی داری فایل رو نگاه می‌کنی. خیلی از پکیج‌ها برای نمایش خروجی از اون استفاده می‌کنن.
- Libcap این پکیج رابط‌های فضای کاربری برای توانایی‌های POSIX 1003.1e که در هسته‌های لینوکس موجوده رو پیاده‌سازی می‌کنه.
- Libelf پروژه elfutils کتابخانه‌ها و ابزارهایی برای فایل‌های ELF و داده‌های DWARF فراهم می‌کنه. اکثر ابزارهای این پکیج در پکیج‌های دیگه موجوده، ولی کتابخانه برای ساخت هسته لینوکس با پیکربندی پیش‌فرض (و کارآمدترین) لازمه.
- Libffi این پکیج یک رابط برنامه‌نویسی سطح بالا و قابل حمل به انواع قراردادهای فراخوانی پیاده‌سازی می‌کنه. بعضی برنامه‌ها ممکنه هنگام کامپایل ندونن چه آرگومان‌هایی باید به یک تابع داده بشه. مثلاً، یک مفسر ممکنه هنگام اجرا در مورد تعداد و انواع آرگومان‌هایی که برای فراخوانی یک تابع استفاده می‌شه، اطلاع داده بشه.
- Libffi می‌تونه در چنین برنامه‌هایی برای ایجاد یک پل از برنامه مفسر به کد کامپایل شده استفاده بشه.
- Libpipeline پکیج Libpipeline یک کتابخانه برای دستکاری خطوط لوله از زیرفرآیندها به روشی انعطاف‌پذیر و راحت فراهم می‌کنه. برای پکیج Man-DB لازمه
- Libtool این پکیج اسکریپت پشتیبانی کتابخانه عمومی GNU رو داره. اون پیچیدگی استفاده از کتابخانه‌های مشترک رو در یک رابط سازگار و قابل حمل جمع می‌کنه. برای مجموعه‌های تست در پکیج‌های دیگه LFS لازمه
- Libxcrypt این پکیج کتابخانه libcrypt رو فراهم می‌کنه که برای هش کردن رمزهای عبور توسط پکیج‌های مختلف (به ویژه Shadow) لازمه. اون جایگزین پیاده‌سازی منسوخ شده libcrypt در Glibc می‌شه

- Linux Kernel این پکیج سیستم عامله. همون لینوکس توی محیط GNU/Linux هست.
- M4 این پکیج یک پردازنده کلی ماکرو متنی فراهم می‌کنه که به عنوان یک ابزار ساخت برای برنامه‌های دیگه مفیده
- Make این پکیج برنامه‌ای داره که کار ساخت پکیج‌ها رو هدایت می‌کنه. تقریباً برای همه پکیج‌های LFS لازمه
- Man-DB این پکیج برنامه‌هایی برای پیدا کردن و دیدن صفحات راهنما ((man pages داره. به خاطر قابلیت‌های بین‌المللی‌سازی بهترش به جای پکیج man انتخاب شده.
- Man-pages این پکیج محتوای اصلی صفحات راهنمای پایه لینوکس رو فراهم می‌کنه.
- Meson این پکیج یک ابزار نرم‌افزاری برای خودکارسازی ساخت نرم‌افزار داره. هدف اصلی Meson کم کردن زمانیه که توسعه‌دهندگان نرم‌افزار باید صرف پیکربندی یک سیستم ساخت کنن. برای ساخت Systemd و خیلی از پکیج‌های BLFS لازمه
- MPC این پکیج توابع حسابی برای اعداد مختلط فراهم می‌کنه. برای GCC لازمه
- MPFR این پکیج توابعی برای حساب دقیق چندین رقمی داره. برای GCC لازمه
- Ninja این پکیج یک سیستم ساخت کوچک با تمرکز روی سرعت داره. طراحی شده که فایل‌های ورودی‌اش توسط یک سیستم ساخت سطح بالاتر تولید بشه و ساخت‌ها رو تا جای ممکن سریع انجام بده. برای Meson لازمه.
- Ncurses این پکیج کتابخانه‌هایی برای کنترل مستقل از ترمینال صفحات کاراکتری داره. اغلب برای فراهم کردن کنترل کرسر برای یک سیستم منو استفاده میشه. برای تعدادی از پکیج‌های LFS لازمه.

- Openssl این پکیج ابزارهای مدیریتی و کتابخانه‌های مربوط به رمزنگاری داره. این‌ها توابع رمزنگاری رو به پکیج‌های دیگه، از جمله هسته لینوکس، فراهم می‌کنن
- Patch این پکیج برنامه‌ای برای تغییر دادن یا ایجاد فایل‌ها با اعمال یک فایل پچ که معمولاً توسط برنامه diff ساخته میشه، داره. برای فرآیند ساخت چندین پکیج LFS لازمه. Perl این پکیج یک مفسر برای زبان اجرایی PERL داره. برای نصب و مجموعه‌های تست چندین پکیج LFS لازمه
- Pkgconf این پکیج برنامه‌ای داره که به پیکربندی پرچم‌های کامپایلر و لینکر برای کتابخانه‌های توسعه کمک می‌کنه. این برنامه می‌تونه به جای pkg-config استفاده بشه، که برای سیستم ساخت خیلی از پکیج‌ها لازمه. این پکیج فعال‌تر و کمی سریع‌تر از پکیج اصلی Pkg-config نگهداری میشه
- Procps-NG این پکیج برنامه‌هایی برای نظارت بر فرآیندها داره. این برنامه‌ها برای مدیریت سیستم مفیدن و همچنین توسط اسکریپت‌های بوت LFS استفاده میشن.
- Psmisc این پکیج برنامه‌هایی برای نمایش اطلاعات درباره فرآیندهای در حال اجرا داره. این برنامه‌ها برای مدیریت سیستم مفیدن.
- Python این پکیج یک زبان برنامه‌نویسی تفسیری داره که فلسفه طراحی‌اش تاکید روی خوانایی کد داره
- Readline این پکیج مجموعه‌ای از کتابخانه‌هاست که قابلیت‌های ویرایش خط فرمان و تاریخچه رو ارائه می‌ده. توسط Bash استفاده میشه.
- Sed این پکیج بهت اجازه می‌ده متن‌ها رو بدون باز کردن در یک ویرایشگر متن ویرایش کنی. برای اسکریپت‌های پیکربندی خیلی از پکیج‌های LFS هم لازمه.
- Shadow این پکیج برنامه‌هایی برای امنیتی کردن رمزهای عبور داره
- Sysklogd این پکیج برنامه‌هایی برای ثبت پیام‌های سیستمی مثل اون‌هایی که توسط هسته یا فرآیندهای دیمن در زمان رویدادهای غیرمعمول منتشر می‌شن، داره.

- Sysvinit این پکیج برنامه init رو فراهم می‌کنه که والد همه فرآیندهای دیگه در یک سیستم لینوکس در حال اجراست
- Udev این پکیج یک مدیر دستگاهه. به صورت پویا مالکیت، دسترسی‌ها، نام‌ها و لینک‌های نمادین گره‌های دستگاه در دایرکتوری /dev رو کنترل می‌کنه وقتی دستگاه‌ها به سیستم اضافه یا از اون حذف می‌شن
- Tar این پکیج قابلیت‌های بایگانی و استخراج تقریباً تمام پکیج‌های استفاده شده در LFS رو فراهم می‌کنه
- Tcl این پکیج زبان دستوری ابزار(Tool Command Language) رو داره که در خیلی از مجموعه‌های تست استفاده می‌شه.
- Texinfo این پکیج برنامه‌هایی برای خواندن، نوشتن و تبدیل صفحات اطلاعاتی داره. در روش‌های نصب خیلی از پکیج‌های LFS استفاده می‌شه.
- Util-linux این پکیج برنامه‌های کاربردی متفرقه داره. در میان اون‌ها، ابزارهایی برای مدیریت سیستم‌های فایل، کنسول‌ها، پارتیشن‌ها و پیام‌ها هستن.
- Vim این پکیج یک ویرایشگر داره. به خاطر سازگاری با ویرایشگر کلاسیک vi و تعداد زیادی قابلیت‌های قدرتمندش انتخاب شده. ویرایشگر یک انتخاب خیلی شخصی برای خیلی از کاربرانه. اگه بخوای، می‌تونی هر ویرایشگر دیگه‌ای جایگزین کنی.
- Wheel این پکیج یک ماژول پایتون داره که پیاده‌سازی مرجع استاندارد پکیج‌بندی چرخ پایتونه
- XML::Parser این پکیج یک ماژول پرله که با Expat ارتباط برقرار می‌کنه.
- XZ Utils این پکیج برنامه‌هایی برای فشرده‌سازی و باز کردن فایل‌ها داره. بالاترین سطح فشرده‌سازی که به طور کلی موجوده رو فراهم می‌کنه و برای باز کردن پکیج‌ها در فرمت XZ یا LZMA مفیده
- Zlib این پکیج روتین‌های فشرده‌سازی و باز کردن رو داره که توسط برخی برنامه‌ها استفاده می‌شه.

- Zstd این پکیج روتین‌های فشرده‌سازی و باز کردن رو داره که توسط برخی برنامه‌ها استفاده می‌شه. نسبت‌های فشرده‌سازی بالایی رو فراهم می‌کنه و یک طیف وسیعی از فشرده‌سازی / سرعت رو ارائه می‌دهد.

ساختار این کتاب

این کتاب به این بخشها تقسیم میشود :

بخش **اول**: مقدمه

در بخش اول، چند نکته مهم در مورد نحوه پیش رفتن با نصب LFS توضیح داده شده. این بخش همچنین اطلاعاتی در مورد خود کتاب ارائه میدهد.

بخش **دوم**: آمادهسازی برای ساخت

بخش دوم توضیح میده که چطور برای فرآیند ساخت آماده بشیم—ایجاد یک پارتیشن، دانلود پکیج ها، و کامپایل ابزارهای موقت.

بخش **سوم**: ساختن ابزارهای موقت و Cross Toolchain LFS

بخش سوم دستورالعمل هایی برای ساخت ابزارهای لازم برای ساختن سیستم LFS نهایی رو فراهم میکنه.

بخش **چهارم**: ساخت سیستم LFS

بخش چهارم، خواننده این کتاب رو از طریق ساخت سیستم LFS هدایت میکنه—کامپایل و نصب تمام پکیجها یکی پس از دیگری، راه اندازی اسکریپت های بوت، و نصب کرنل. سیستم لینوکس حاصل، پایه ای است که نرم افزارهای دیگر میتوانند بر روی آن ساخته شوند تا سیستم را به دلخواه گسترش دهند. در انتهای این کتاب، فهرستی آسان برای استفاده وجود دارد که تمام برنامه ها، کتابخانه ها، و فایل های مهم نصب شده را لیست میکند.

بخش پنجم اطلاعاتی در مورد خود کتاب ارائه میده شامل اختصارات و اصطلاحات، تشکر و قدردانی، وابستگی های پکیج(Package dependencies)، لیستی از اسکریپتهای بوت LFS، مجوزهای توزیع کتاب، و فهرست جامعی از پکیج ها، برنامه ها، کتابخانه ها، و اسکریپت ها.

اصلاحات و هشدار های امنیتی

نرم افزار استفاده شده برای ساخت یک سیستم LFS دائماً در حال به روزرسانی و بهبوده. هشدارهای امنیتی و رفع اشکالات(Bug fixes) ممکنه بعد از انتشار کتاب LFS موجود بشن. برای چک کردن اینکه آیا نسخه های پکیج ها یا دستورالعمل ها در این نسخه از LFS به تغییراتی نیاز دارن—برای تعمیر آسیب‌پذیری های امنیتی یا رفع اشکالات دیگه—لطفاً قبل از ادامه دادن به ساخت، به وبسایت زیر سر بزنید :

https://www.linuxfromscratch.org/lfs/errata/12.1

شما باید به تغییرات توجه داشته باشید و اون هارو تو بخش‌های این کتاب درحین ساخت سیستم LFS اعمال کنید و اگر قصد دارید از سیستم LFS به عنوان یک سیستم دسکتاپ یا سرور واقعی استفاده کنید، باید به طور مداوم به هشدارها مراجعه کنید و هرگونه آسیبپذیری امنیتی رو، حتی وقتی سیستم LFS به طور کامل ساخته شده، برطرف کنید.

بخش اول. مقدمه

فصل ۱. مقدمه

۱.۱. چگونه یک سیستم LFS بسازیم

سیستم LFS با استفاده از یک توزیع لینوکس قبلا نصب شده (مثل Debian، Fedora، OpenMandriva، یا openSUSE) ساخته می‌شود. این سیستم لینوکس موجود (میزبان) به عنوان نقطه شروع برای ارائه برنامه‌های لازم، از جمله کامپایلر، لینکر[۹]، و شل، برای ساخت سیستم جدید استفاده می‌شود. در هنگام نصب توزیع، گزینه "development" را انتخاب کنید تا این ابزارها شامل آن شوند.

توجه: راه‌های زیادی برای نصب یک توزیع لینوکس وجود دارد و تنظیمات پیش‌فرض (Default configuration) برای ساخت یک سیستم LFS بهینه نیست.پس برای پیشنهادها مربوطه به راه اندازی یک توزیع lfs به این آدرس مراجعه کنید:
https://www.linuxfromscratch.org/hints/downloads/files/partitioning-for-lfs.txt

به عنوان یک راه جایگزین برای نصب یک توزیع جداگانه روی دستگاهتون، ممکنه بخواهید از یک LiveCD از یک توزیع تجاری استفاده کنید.

فصل ۲ این کتاب یاد میگیرید چطوری یک پارتیشن و فایل سیستم جدید لینوکس native ایجاد کنید و در ادامه سیستم LFS جدیدتون را کامپایل و نصب کنید.

فصل ۳ توضیح می‌دهد کدام پکیج ها و پچها باید برای ساخت یک سیستم LFS دانلود، و چگونه آن‌ها را در فایل سیستم جدید نصب کنید.

فصل ۴ در مورد راه اندازی یک محیط کار مناسب بحث می‌کنیم. لطفا فصل ۴ را با دقت بخوانید زیرا چندین مسئله مهمی را که باید قبل از شروع کار خود از فصل ۵ به بعد از آن آگاه باشید، توضیح می‌دهد.

فصل ۵ نصب اولیه toolchain[۱۰] را توضیح می‌دهد، (gcc، binutils، و glibc) با استفاده از تکنیک‌های کامپایل متقاطع(cross-compilation) برای جدا کردن ابزارهای تازه ساخته شده از سیستم میزبان(سیستمی که ازش برای ساخت lfs استفاده میکنید).

---

۹. لینکر یک برنامه در سیستم است که یک یا چند فایل ابجکت (که توسط کامپایلر یا اسمبلر ایجاد شده) را گرفته و آن‌ها را به یک فایل اجرایی(executable)، فایل کتابخانه یا فایل "ابجکت" دیگر تبدیل می‌کند.

۱۰. toolchain در توسعه نرم‌افزار مجموعه‌ای از ابزارهای برنامه‌نویسی است که با هم برای انجام یک وظیفه توسعه نرم‌افزار پیچیده یا ایجاد یک محصول نرم‌افزاری استفاده می‌شوند. این معمولا یک برنامه کامپیوتری دیگر یا مجموعه‌ای از برنامه‌های مرتبط است.

فصل ۶ به شما نشان می‌دهد چگونه با استفاده از cross-toolchain تازه ساخته شده، ابزارهای اولیه را cross-compile[۱۱] کنید.

. **فصل ۷** سپس وارد یک محیط "chroot" می‌شود، جایی که ما از ابزارهای جدید برای ساخت تمام بقیه ابزارهای لازم برای ایجاد سیستم LFS استفاده می‌کنیم. این تلاش برای جدا کردن سیستم جدید از توزیع میزبان ممکن است بیش از حد به نظر برسد. توضیح فنی کامل در مورد اینکه چرا این کار انجام می‌شود در Toolchain Technical Notes ارائه شده است.

در **فصل ۸** سیستم LFS کامل ساخته می‌شود. مزیت دیگری که محیط chroot ارائه می‌دهد این است که به شما اجازه می‌دهد تا در حالی که LFS در حال ساخت است، از سیستم میزبان خود استفاده کنید. در حالی که منتظر تکمیل کامپایل بسته‌ها هستید، می‌توانید همانطور که معمولا استفاده می‌کنید، از کامپیوتر خود استفاده کنید. برای اتمام نصب، پیکربندی اولیه سیستم در **فصل ۹** تنظیم میشود، و کرنل و بوت لودر در **فصل ۱۰** ایجاد می‌شوند.

فصل ۱۱ شامل اطلاعاتی در مورد ادامه تجربه LFS فراتر از این کتاب است. پس از اجرای مراحل در این فصل، کامپیوتر آماده برای بوت شدن به سیستم LFS جدید است. این فرآیند فقط به طور خلاصه است و. اطلاعات دقیق در مورد هر گام در فصل‌های مربوطه ارائه شده است. مواردی که اکنون پیچیده به نظر می‌رسند، توضیح داده خواهند شد، و همه چیز سرجای خود قرار خواهد گرفت هنگامی که شما ماجراجویی LFS خود را شروع می‌کنید.

---

۱۱. "cross-compile" به معنی استفاده از یک کامپایلر روی یک پلتفرم (مثلاً ویندوز) برای تولید کد قابل اجرا برای یک پلتفرم دیگر (مثلاً اندروید) است.برای مثال، اگر شما برنامه‌ای را برای اجرا بر روی یک دستگاه اندروید بنویسید ولی کد شما را بر روی یک کامپیوتر شخصی کامپایل کنید، کامپایلری که استفاده می‌کنید یک کامپایلر متقاطع یا همان cross-compiler است.