

LPIC-2
(201-450)
(202-450)



LPIC-2 REFERENCE

Study Guide ^{1st} Edition

معرفی کوتاه کتاب

سطح دوم از سری گواهینامه‌های حرفه‌ای لینوکس تحت نظر سازمان (Linux Professional Institute) LPI است. این گواهینامه تخصصی برای کسانی طراحی شده است که پیش از این مهارت‌های پایه‌ای لینوکس را بدست آورده‌اند (با گذراندن LPIC-1) و اکنون می‌خواهند دانش خود را در زمینه اداره شبکه‌ها، خدمات شبکه، امنیت و نگهداری سیستم‌ها تقویت کنند. در واقع، LPIC-2 مرز بین مدیر سیستم محلی و متخصص سیستم‌های درون شبکه را نشان می‌دهد. باید بتواند شبکه‌ای ترکیبی از لینوکس و میکروسافت را برنامه‌ریزی، پیاده‌سازی، حفظ کند و در شرایط بحرانی عیب‌یابی نماید. او باید بتواند سرویس‌های مهم شبکه را راه‌اندازی کند — مانند سرور داخلی با سرویس‌هایی نظیر Samba، NFS، DNS و DHCP؛ نقطه دسترسی به اینترنت با فایروال، SSH، VPN، پروکسی وب و سرویس ایمیل؛ و همچنین سرورهای خارجی شامل وب‌سرور با پروکسی معکوس و سرور FTP. علاوه بر آن، داوطلب باید توانایی نظارت بر دستیاران را داشته باشد

کتاب

مرجع

LPIC1

لینوکس

کتاب مرجع کامل مدرک بین المللی

LPIC-2

(201-450, 202-450)

Linux Professional Institute Certification Study Guide

حسین سیلانی

ویرایش اول

۱۴۰۴



سرشناسه	سیلانی، حسین، ۱۳۶۰.
عنوان و نام پدیدآور	Lpic2
مشخصات نشر	حسین سیلانی؛ ویراستاری حسین سیلانی.
مشخصات ظاهری	تهران: کیان دانش، ۱۴۰۴.
شابک	600 ص.
وضعیت فهرست نویسی	۹۷۸.۶۲۲.۴۰۱.۰۵۵.۶
موضوع	فیپا
شناسه افزوده	سیستم عامل لینوکس
رده بندی کنگره	Operating System
رده بندی دیویی	سیلانی، حسین، ۱۳۶۰.
شماره کتابشناسی ملی	۷۶/۷۶QA
اطلاعات رکورد کتابشناسی	۴۳۲/۰۰۵
	۹۶۷۳۵۹۳
	فیپا

نشر کیان دانش: ناشر کتاب‌های دانشگاهی



عنوان: مفاهیم سیستم عامل
تالیف: حسین سیلانی
صفحه آرای و ویراستاری: حسین سیلانی
چاپ و صحافی: موسسه مهراد
طراحی جلد: حسین سیلانی
چاپ اول: ۱۴۰۴
شمارگان: ۱۰۰ جلد
شابک: ۹۷۸.۶۲۲.۴۰۱.۰۵۵.۶
حق چاپ محفوظ است.

خیابان ۱۲ فروردین، خیابان شهدای ژاندارمری، شماره ۱۲۶، طبقه چهارم

سخنی از نویسنده
با سلام و احترام به دوست گرامی

از اینکه این کتاب را انتخاب کرده‌اید و در این مسیر همراه من هستید صمیمانه سپاسگزارم. امیدوارم این اثر برای علاقه‌مندان به حوزه لینوکس مفید و اثربخش باشد. همچنین از شما درخواست دارم که اگر نواقص یا کاستی‌هایی در کتاب مشاهده کردید آن‌ها را نادیده نگیرید و با ارائه نظرات و پیشنهادات سازنده‌ی خود، به بهبود کیفیت این اثر کمک کنید.

شایان ذکر است که این کتاب یکی از صدها کتاب لینوکسی من است که بسیاری از آن‌ها در حال ترجمه یا نگارش هستند. این مجموعه در راستای پروژه‌های متن‌باز شخصی‌ام و با هدف طراحی و توسعه توزیع‌های لینوکس نوشته شده است. امیدوارم این تلاش‌ها گامی هرچند کوچک در جهت گسترش دانش و استفاده از سیستم‌عامل‌های متن‌باز باشد.

🔗 <https://predator.os.ir/>

🔗 <https://littlePsycho.ir>

🔗 <https://emperor.os.ir/>

🔗 <https://hubuntu.ir>

تصمیم دارم محتوای آموزشی فارسی در حوزه‌های لینوکس و نرم‌افزارهای متن‌باز را در قالب‌های متنوعی مانند کتابچه، کتاب، فلش کارت، فایل‌های صوتی و ویدئوهای آموزشی تولید کرده و از طریق یک سایت در دسترس عموم قرار دهم. هدف من از این کار، ایجاد یکی از بزرگترین و جامع‌ترین مراجع آموزشی فارسی در این زمینه است. برای دسترسی به سایت و اطلاع از آخرین اخبار و محتواهای آموزشی، می‌توانید از طریق راه‌های ارتباطی زیر با من در تماس باشید:

🔗 learninghive.ir: آدرس سایت

🔗 [@LinuxTnt](https://t.me/LinuxTnt): کانال تلگرام

🔗 [@seilany](https://t.me/seilany): آیدی تلگرام

🔗 h.seilany@gmail.com: ایمیل

🔗 [hosseinseilani](#): گیت‌هاب

منتظر حضور و همراهی شما هستم!

تقديم به تو

يگانه فرزانه دلم

- ۳۷ - مخاطبین هدف این دوره چه کسانی هستند؟

- ۳۹ - تاریخچه سیستم عامل لینوکس

- ۳۹ - تاریخچه سیستم عامل یونیکس

- ۳۹ - از یونیکس تا لینوکس — روایتی از تولد یک جنبش نرم افزاری آزاد

- ۳۹ - آغاز ماجرا؛ تولد یونیکس ۱۹۶۹

- ۳۹ - زبان C و تحولی بزرگ ۱۹۷۳

- ۳۹ - گسترش در دانشگاه ها و شکل گیری شاخه ها

- ۴۰ - ظهور مینیکس — پلی میان آموزش و عمل ۱۹۸۷

- ۴۰ - ریچارد استالمن و پروژه گنو ۱۹۸۳

- ۴۱ - تولد لینوکس ۱۹۹۱

- ۴۱ - از یک هسته تا یک اکوسیستم جهانی

- ۴۱ - چرا لینوکس ماندگار شد؟

- ۴۱ - نسخه های اولیه یونیکس

- ۴۲ - گسترش و نفوذ در دانشگاه ها

- ۴۳ - انشعابات مهم یونیکس

- ۴۳ - پیدایش مینیکس و مقدمه ای بر لینوکس

- ۴۴ - پروژه گنو و جنبش نرم افزار آزاد

- ۴۵ - تولد لینوکس

- ۴۷ - آیا یادگیری لینوکس سخت است؟

- ۴۸ - کپی رایت و مجوز لینوکس

- ۴۸ - اصول و ویژگی های مجوز GPL

- ۵۱ - فصل ۲

Error! Bookmark not defined. - نخستین توزیع های لینوکس

- ۵۲ - فرایند بوت لینوکس (Linux Boot Process)

- ۵۲ - مشاهده فرایند بوت (Viewing the Boot Process)

- ۵۳ - راه اندازی Firmware میان افزار

- ۵۴ - راه اندازی BIOS

- ۵۴ - محل بارگذاری Bootloader توسط BIOS

- ۵۴ - MBR

- ۵۵ - ساختار MBR

- ۵۵ - عملکرد MBR در فرایند بوت

- ۵۵ - زنجیره ای کردن بوت لودرها (Chainloading)

۵۵ -	راه اندازی (The UEFI Startup) UEFI
۵۵ -	استفاده از ESP – EFI System Partition
۵۶ -	UEFI Boot Manager مدیر بوت UEFI
۵۶ -	پشتیبانی از SSD و Nvme
۵۶ -	Linux Bootloaders بوت لودرهای لینوکس
۵۶ -	LILO – Linux Loader
۵۷ -	GRUB Legacy
۵۷ -	ویژگی های اصلی GRUB Legacy
۵۷ -	محدودیت های GRUB Legacy
۵۷ -	GRUB2
۵۷ -	ویژگی های کلیدی GRUB2
۵۸ -	پوشش بوت لودرهای مدرن
۵۸ -	GRUB Legacy گراب لِگسی
۵۸ -	پیکربندی گراب لِگسی Configuring GRUB Legacy
۵۹ -	تعاریف عمومی
۵۹ -	تعاریف بوت سیستم عامل ها
۶۰ -	نمونه گراب Red Hat
۶۰ -	نمونه برای Debian / Ubuntu سیستم های قدیمی تر با (GRUB Legacy)
۶۰ -	Addressing Drives in GRUB Legacy
۶۰ -	آدرس دهی دیسک ها در گراب لِگسی
۶۱ -	initrd دستور initrd Command
۶۱ -	نصب گراب لِگسی GRUB Legacy
۶۱ -	نصب روی (MBR رایج ترین و پیشنهادی)
۶۱ -	نصب با فرمت GRUB Legacy
۶۱ -	نصب در Boot Sector پارتیشن (برای Chainloading)
۶۲ -	تعامل با گراب لِگسی
۶۲ -	ویرایش گزینه های بوت به صورت لحظه ای در GRUB Legacy
۶۳ -	GRUB2
۶۳ -	پیکربندی GRUB2
۶۳ -	نمونه فایل پیکربندی GRUB2
۶۴ -	استفاده از دستور set
۶۴ -	استفاده از Environment Variables
۶۴ -	سیستم شماره گذاری پارتیشن ها در GRUB2

حذف برخی دستورات قدیمی	۶۴ -
نمونه فایل پیکربندی GRUB2	۶۴ -
فایل‌های پیکربندی در GRUB2	۶۶ -
تنظیمات عمومی در GRUB2	۶۶ -
ایجاد خودکار فایل grub.cfg	۶۶ -
نصب Installing GRUB2	۶۷ -
بازسازی فایل پیکربندی	۶۷ -
تعامل با GRUB2	۶۸ -
بوت‌لودرهای جایگزین	۶۸ -
Systemd-boot	۶۸ -
U-Boot	۶۸ -
Syslinux Project پروژه سیس‌لینوکس	۶۸ -
ISOLINUX	۶۹ -
PXELINUX	۶۹ -
بوت‌لودرهای امن	۶۹ -
روش‌های اجرای لینوکس در محیط Secure Boot	۶۹ -
Mini-Bootloaders مینی بوت‌لودرها	۷۱ -
Process Initialization فرآیند راه‌اندازی	۷۱ -
مسیرهای جستجوی init	۷۱ -
وظیفه اصلی init	۷۱ -
روش‌های متداول Initialization در لینوکس	۷۱ -
SysV Initialization	۷۲ -
نقش مدیر سیستم در SysV	۷۲ -
محدودیت‌های SysV	۷۲ -
Systemd Initialization با Systemd راه‌اندازی	۷۲ -
Upstart Initialization با Upstart راه‌اندازی	۷۲ -
مقایسه روش‌های Initialization	۷۲ -
روش (SysV)	۷۳ -
Runlevels سطوح اجرایی در لینوکس	۷۳ -
اجرای برنامه‌ها در سطوح اجرایی	۷۳ -
روش دوم: Startup Scripts اسکریپت‌های راه‌اندازی	۷۴ -
نام‌گذاری اسکریپت‌ها:	۷۴ -
تغییر سطوح اجرایی برنامه‌ها	۷۴ -

۷۵ -	chkconfig
۷۵ -	فرمت‌های دستور (chkconfig)
۷۷ -	بررسی سطح اجرایی سیستم
۷۷ -	تغییر سطوح اجرایی
۷۷ -	روشن شدن systemd
۷۸ -	Units and Targets واحدها و اهداف
۷۹ -	پیکربندی واحدها (Configuring Units)
۸۰ -	فایل‌های پیکربندی Target
۸۰ -	تنظیم Target پیش فرض (Setting the Default Target)
۸۰ -	برنامه systemctl و مدیریت سرویس‌ها در لینوکس
۸۲ -	بررسی وضعیت سرویس‌ها با systemctl
۸۲ -	تغییر Target ها و سطح اجرای سیستم
۸۲ -	بارگذاری مجدد و راه‌اندازی سرویس‌ها
۸۳ -	فعال‌سازی و غیرفعال‌سازی سرویس‌ها در هنگام بوت
۸۳ -	بررسی تغییرات فایل‌های پیکربندی با systemd-delta
۸۳ -	مدیریت لاگ‌ها در systemd
۸۴ -	روشن شدن Upstart در لینوکس
۸۴ -	جایگزینی SysV init با Upstart
۸۵ -	نمونه‌ای از یک فایل پیکربندی Upstart
۸۵ -	ویژگی‌های مهم Upstart
۸۵ -	تغییر تنظیمات سرویس‌ها در Upstart
۸۶ -	مدیریت سرویس‌ها با دستورات Upstart
۸۶ -	سازگاری Upstart با LSB (Linux Standard Base)
۸۷ -	بازیابی سیستم در لینوکس (System Recovery)
۸۷ -	۱. خطاهای کرنل (Kernel Failures)
۸۷ -	انتخاب کرنل قبلی هنگام بوت
۸۷ -	حالت تک‌کاربره (Single-User Mode)
۸۸ -	اضافه کردن پارامترهای کرنل (Passing Kernel Parameters)
۸۸ -	۲. خرابی دیسک (Root Drive Failure)
۸۹ -	استفاده از Rescue Disk دیسک نجات
۸۹ -	ابزار اصلی: دستور fsck
۸۹ -	اجرای fsck روی پارتیشن Root

- ۹۰ - ----- fsck چند بار اجرای
- ۹۰ - ----- Mount کردن Root Drive پس از تعمیر
- ۹۰ - ----- جدا کردن (Unmount) پارتیشن
- ۹۰ - ----- ریپوت کردن سیستم
- ۹۰ - ----- نکات کلیدی (Essentials):

فصل ۳ ----- ۹۲ - نگهداری و پایداری سیستم لینوکس ----- ۹۲ -

- ۹۲ - ----- اطلاع‌رسانی به کاربران
- ۹۲ - ----- پیام‌رسانی لحظه‌ای (Fluid Messaging)
- ۹۲ - ----- دستور wall
- ۹۳ - ----- مدیریت پیام‌ها با msg
- ۹۳ - ----- پشت‌صحنه msg
- ۹۳ - ----- اهمیت اطلاع‌رسانی درست
- ۹۴ - ----- دستور write در لینوکس
- ۹۴ - ----- بررسی دسترسی پیام با دستور -T who
- ۹۵ - ----- ارسال پیام با write
- ۹۵ - ----- دستور wall و پیام‌همگانی
- ۹۵ - ----- ارتباط wall با systemctl
- ۹۶ - ----- پیام‌های wall و تجربه کاربر
- ۹۶ - ----- دسترسی و امنیت wall
- ۹۶ - ----- نکات عملی و سناریوهای واقعی
- ۹۷ - ----- استفاده از دستور -send notify
- ۹۷ - ----- ساختار و نحوه استفاده
- ۹۸ - ----- ارسال پیام به کاربران دیگر با GUI
- ۹۹ - ----- اتوماسیون پیام‌های -send notify
- ۱۰۰ - ----- مقایسه wall و -send notify
- ۱۰۰ - ----- استفاده از دستور /shutdown/sbin
- ۱۰۱ - ----- پارامتر زمان (time)
- ۱۰۱ - ----- پارامتر پیام ([wall message])
- ۱۰۱ - ----- لغو shutdown
- ۱۰۱ - ----- گزینه‌های پیشرفته shutdown
- ۱۰۲ - ----- اطلاع‌رسانی فعال (Fluid Messaging)

- ۱۰۲ - اطلاع‌رسانی ایستا (Static Messaging)
- ۱۰۳ - استفاده از فایل `etc/issue`
- ۱۰۳ - ویرایش فایل `etc/issue` برای اطلاع‌رسانی
- ۱۰۳ - اطلاع‌رسانی برای کاربران GUI
- ۱۰۴ - استفاده از فایل `etc/issue.net`
- ۱۰۴ - فعال‌سازی `etc/issue.net` در OpenSSH
- ۱۰۴ - نکات تکمیلی `etc/issue` و `etc/issue.net`
- ۱۰۵ - استفاده از فایل `etc/motd`
- ۱۰۵ - کاربردهای سنتی و مدرن فایل `etc/motd`
- ۱۰۵ - ساخت و ویرایش فایل `etc/motd`
- ۱۰۵ - نمایش پویا و توزیع‌های مختلف
- ۱۰۶ - مقایسه پیام‌های ایستا و پویا
- ۱۰۶ - نکات مهم برای استفاده از `etc/motd`
- ۱۰۶ - پشتیبان‌گیری از سیستم
- ۱۰۶ - توسعه یک استراتژی پشتیبان‌گیری
- ۱۰۷ - عوامل مهم در طراحی برنامه پشتیبان‌گیری
- ۱۰۷ - انتخاب رسانه‌های پشتیبان‌گیری
- ۱۰۷ - ۱. نوار مغناطیسی (Magnetic Tape)
- ۱۰۸ - ۲. دیسک‌های نوری (Optical Discs)
- ۱۰۸ - ۳. هارد دیسک‌ها (HDD)
- ۱۰۸ - ۴. درایوهای حالت جامد (SSD)
- ۱۰۸ - بررسی راهکارهای ذخیره‌سازی و چرخش پشتیبان‌گیری
- ۱۰۹ - پشتیبان‌گیری از راه دور
- ۱۰۹ - پشتیبان‌گیری ابری
- ۱۰۹ - بررسی انواع پشتیبان‌گیری
- ۱۱۰ - بررسی بازیابی داده‌ها
- ۱۱۰ - بازیابی فایل‌های منفرد یا چندگانه
- ۱۱۰ - بازیابی پارتیشن یا دیسک کامل
- ۱۱۰ - بازیابی سیستم
- ۱۱۱ - بررسی دایرکتوری‌های مورد نیاز برای پشتیبان‌گیری
- ۱۱۱ - مستندسازی بسته‌های نصب شده
- ۱۱۱ - دایرکتوری‌های کلیدی برای پشتیبان‌گیری

- ۱۱۲ - توجه به نسخه‌های لینوکس
- ۱۱۲ - بررسی راهکارهای نرم‌افزاری پشتیبان‌گیری
- ۱۱۲ - Amanda
- ۱۱۲ - Bacula
- ۱۱۳ - Bareos
- ۱۱۳ - Duplicity
- ۱۱۳ - BackupPC
- ۱۱۳ - ابزارهای خط فرمان برای پشتیبان‌گیری
- ۱۱۴ - اجرای پشتیبان‌گیری
- ۱۱۴ - استفاده از tar برای انجام پشتیبان‌گیری
- ۱۱۴ - مفهوم آرشیو و tarball
- ۱۱۴ - گزینه‌های پرکاربرد tar برای پشتیبان‌گیری
- ۱۱۵ - پشتیبان‌گیری کامل و افزایشی با tar
- ۱۱۵ - اعتبارسنجی و بررسی پشتیبان‌ها
- ۱۱۶ - بازیابی داده‌ها از آرشیو
- ۱۱۷ - کار با نوارهای مغناطیسی
- ۱۱۷ - ساختار و مارکرهای نوار
- ۱۱۷ - درایوهای نوار SCSI
- ۱۱۸ - مدیریت نوار با دستور mt
- ۱۱۸ - عملیات متداول mt
- ۱۱۸ - پشتیبان‌گیری با tar روی نوار
- ۱۱۹ - بازیابی داده‌ها از نوار
- ۱۱۹ - استفاده از rsync برای پشتیبان‌گیری
- ۱۱۹ - پشتیبان‌گیری محلی با rsync
- ۱۲۰ - پشتیبان‌گیری از طریق شبکه با rsync
- ۱۲۰ - استفاده پیشرفته از rsync
- ۱۲۰ - استفاده از dd برای کپی داده‌ها
- ۱۲۰ - دستور پایه dd
- ۱۲۱ - پاک کردن کامل یک دیسک با dd
- ۱۲۱ - گزینه‌های پیشرفته dd
- ۱۲۱ - نصب برنامه‌ها از سورس (Source)
- ۱۲۲ - مراحل کلی نصب برنامه از سورس

دریافت فایل‌های نصب	۱۲۲ -
باز کردن بسته نصب (Unpacking)	۱۲۲ -
مطالعه مستندات نصب	۱۲۳ -
آماده‌سازی برای کامپایل	۱۲۳ -
کامپایل برنامه	۱۲۳ -
تکمیل نصب	۱۲۳ -
مدیریت استفاده از منابع سیستم (Managing Resource Usage)	۱۲۵ -
اندازه‌گیری استفاده از منابع (Measuring Resource Usage)	۱۲۵ -
اهمیت اندازه‌گیری منابع	۱۲۵ -
منابع مهم برای پایش	۱۲۵ -
ابزارهای خط فرمان برای پایش منابع	۱۲۶ -
نکات عملی استفاده از ابزارهای پایش	۱۲۶ -
ابزار sar	۱۲۷ -
مجموعه ابزارهای پیشرفته برای پایش مصرف منابع	۱۲۷ -
نحوه کارکرد sar	۱۲۸ -
نقش sdc	۱۲۸ -
پیش‌بینی مصرف منابع (Predicting Resource Usage)	۱۲۹ -
نرم‌افزارهای جامع پایش منابع	۱۲۹ -
1. Cacti	۱۲۹ -
2. collectd	۱۲۹ -
3. MRTG (Multi Router Traffic Grapher)	۱۳۰ -
4. Nagios	۱۳۰ -
5. Icinga	۱۳۰ -
6. RRDTool	۱۳۰ -
عیب‌یابی مصرف منابع در لینوکس (Troubleshooting Resource Usage)	۱۳۱ -
حافظه (Memory / RAM)	۱۳۱ -
ابزارهای پایش حافظه	۱۳۱ -
فرآیندها (Processes)	۱۳۲ -
پردازنده (CPU)	۱۳۲ -
ورودی / خروجی دستگاه‌ها (Device I/O)	۱۳۳ -
بهنای باند شبکه (Network Throughput)	۱۳۳ -
همبستگی بین منابع	۱۳۴ -
هسته لینوکس (Linux Kernel)	۱۳۸ -

- ۱۳۹ - هسته چیست؟
- ۱۳۹ - ویژگی‌های هسته لینوکس
- ۱۳۹ - چهاروظیفه اصلی هسته لینوکس
- ۱۴۰ - مدیریت حافظه سیستم (System Memory Management)
- ۱۴۰ - مفهوم Swap Space
- ۱۴۰ - صفحات حافظه (Memory Pages)
- ۱۴۰ - نکات کلیدی مدیریت حافظه
- ۱۴۱ - حافظه خصوصی و حافظه اشتراکی
- ۱۴۲ - مدیریت برنامه‌ها و فرآیندها در لینوکس
- ۱۴۲ - فرآیند init
- ۱۴۲ - سطوح اجرای (Run Levels)
- ۱۴۲ - پایش و مدیریت حافظه در عمل
- ۱۴۳ - مدیریت سیستم و فرآیندها در لینوکس
- ۱۴۳ - مشاهده فرآیندهای در حال اجرا با دستور ps
- ۱۴۳ - مدیریت سخت‌افزار توسط هسته لینوکس
- ۱۴۴ - مزیت ماژول‌های هسته
- ۱۴۴ - فایل‌های دستگاه در لینوکس
- ۱۴۵ - مدیریت فایل سیستم در لینوکس
- ۱۴۶ - رابط بین هسته و فایل سیستمها Virtual File System
- ۱۴۷ - ویژگی‌های عملی فایل سیستمها
- ۱۴۷ - مدیریت فایل سیستم با ابزارهای لینوکس
- ۱۴۸ - بخش‌های مختلف هسته لینوکس (Linux Kernel)
- ۱۴۸ - فایل باینری هسته (Kernel Binary)
- ۱۴۸ - ماژول‌های هسته (Kernel Modules)
- ۱۴۹ - کد منبع هسته (Kernel Source)
- ۱۴۹ - پچ‌های هسته (Kernel Patches)
- ۱۴۹ - هدرهای هسته (Kernel Headers)
- ۱۵۰ - مستندات هسته (Kernel Documentation)
- ۱۵۰ - نسخه‌های هسته لینوکس (Kernel Versions)
- ۱۵۰ - نسخه‌های اولیه هسته
- ۱۵۰ - سری نسخه‌های ۱ لینوکس
- ۱۵۱ - سری نسخه‌های ۲ لینوکس
- ۱۵۱ - سری نسخه‌های ۲.۶ لینوکس

- ۱۵۱ - _____ سری نسخه‌های ۳ لینوکس
- ۱۵۲ - _____ سری نسخه‌های ۴ لینوکس
- ۱۵۲ - _____ کامپایل هسته لینوکس (Compiling a Kernel)
- ۱۵۲ - _____ نصب هسته جدید (Kernel Binary)
- ۱۵۳ - _____ دریافت کد منبع (Obtaining Source Code)
- ۱۵۴ - _____ ایجاد فایل پیکربندی هسته لینوکس (Creating the Kernel Configuration File)
- ۱۵۴ - _____ ایجاد خودکار فایل پیکربندی
- ۱۵۵ - _____ استفاده از منوی گرافیکی xconfig
- ۱۵۵ - _____ پاکسازی پیکربندی قدیمی
- ۱۵۶ - _____ ذخیره فایل پیکربندی
- ۱۵۷ - _____ کامپایل و نصب هسته لینوکس (Compiling and Installing the Kernel)
- ۱۵۷ - _____ پاکسازی فایل‌های قدیمی قبل از کامپایل
- ۱۵۷ - _____ کامپایل فایل باینری هسته
- ۱۵۸ - _____ نصب فایل باینری هسته
- ۱۵۸ - _____ فایل System.map
- ۱۵۸ - _____ نصب خودکار هسته با make
- ۱۵۸ - _____ کامپایل و نصب ماژول‌ها
- ۱۵۹ - _____ ایجاد دیسک RAM اولیه (Initial RAM Disk)
- ۱۵۹ - _____ ابزار mkinitrd و ایجاد Initial RAM Disk
- ۱۵۹ - _____ فرمت عمومی دستور mkinitrd
- ۱۵۹ - _____ گزینه‌های خط فرمان mkinitrd
- ۱۶۱ - _____ اهمیت Initial RAM Disk
- ۱۶۱ - _____ محل ذخیره فایل Initial RAM Disk
- ۱۶۲ - _____ ابزار mkinitramfs و ایجاد Initial RAM Disk در توزیع‌های Debian
- ۱۶۲ - _____ فرمت عمومی دستور mkinitramfs
- ۱۶۲ - _____ گزینه‌های خط فرمان mkinitramfs
- ۱۶۳ - _____ محل ذخیره فایل Initial RAM Disk
- ۱۶۳ - _____ اهمیت mkinitramfs و تفاوت آن با mkinitrd
- ۱۶۴ - _____ بوت کردن هسته جدید لینوکس
- ۱۶۴ - _____ بوت هسته با GRUB Legacy
- ۱۶۴ - _____ بوت هسته با GRUB2
- ۱۶۵ - _____ ایجاد بسته هسته (Kernel Package)
- ۱۶۵ - _____ نگهداری هسته لینوکس (Maintaining the Kernel)

- ۱۶۶ - (Working with Module Files) کار با فایل‌های ماژول
- ۱۶۶ - فایل‌های بارگذاری ماژول‌ها هنگام بوت
- ۱۶۶ - پیکربندی ماژول‌ها
- ۱۶۶ - مدیریت ماژول‌های سفارشی با DKMS
- ۱۶۷ - دستورات مدیریتی ماژول‌ها
- ۱۶۷ - لیست ماژول‌ها: دستور lsmod
- ۱۶۷ - مشاهده اطلاعات ماژول‌ها: دستور modinfo
- ۱۶۷ - نصب ماژول‌ها
- ۱۶۷ - دستور insmod
- ۱۶۸ - دستور modprobe
- ۱۶۸ - حذف ماژول‌ها
- ۱۶۹ - کار با سخت‌افزار در لینوکس (Working with Hardware)
- ۱۶۹ - کار با کارت‌های PCI
- ۱۶۹ - کار با دستگاه‌های USB
- ۱۷۰ - شناسایی خودکار سخت‌افزار
- ۱۷۰ - Udevd
- ۱۷۱ - عیب‌یابی هسته لینوکس (Troubleshooting the Kernel)
- ۱۷۱ - نمایش نسخه هسته لینوکس
- ۱۷۲ - سیستم فایل proc /
- ۱۷۳ - پارامترهای هسته

فصل ۴ - ۱۷۵ -

مدیریت فایل سیستم‌ها در لینوکس - ۱۷۵ -

- ۱۷۵ - عملکرد و مدیریت فایل سیستم لینوکس
- ۱۷۵ - درک ساختار فایل سیستم‌ها
- ۱۷۵ - پارتیشن‌بندی
- ۱۷۶ - فرمت سطح بالا
- ۱۷۶ - متادیتای فایل‌ها و جدول inode
- ۱۷۶ - درک انواع فایل سیستم‌ها
- ۱۷۶ - فایل سیستم‌های Native Linux
- ۱۷۷ - ژورنالینگ در فایل سیستم‌ها
- ۱۷۷ - Copy-On-Write (COW)
- ۱۷۷ - مزایای انتخاب فایل سیستم مناسب
- ۱۷۷ - 1. Btrfs (B-tree File System)
- ۱۷۸ - 2. Ext4 (Fourth Extended File System)
- ۱۷۸ - 3. ReiserFS

- ۱۷۸ - (Non-native Linux Filesystems) بررسی فایل سیستم های غیر بومی لینوکس
- ۱۷۹ - نمونه های رایج فایل سیستم غیر بومی
- ۱۸۰ - مقایسه Native و Non-native Filesystems
- ۱۸۰ - ساخت فایل سیستم در لینوکس
- ۱۸۰ - ابزارهای اصلی ایجاد فایل سیستم
- ۱۸۱ - بررسی نوع فایل سیستم ساخته شده
- ۱۸۲ - اتصال فایل سیستم به ساختار دایرکتوری لینوکس
- ۱۸۲ - اتصال موقت فایل سیستم (Temporary Mounting)
- ۱۸۲ - بررسی فایل سیستم های متصل شده
- ۱۸۳ - گزینه های دستور mount
- ۱۸۳ - فایل سیستم ZFS در لینوکس
- ۱۸۴ - قطع اتصال فایل سیستم در لینوکس
- ۱۸۴ - مشکل اشغال بودن فایل سیستم
- ۱۸۵ - اتصال دستی رسانه های قابل جابجایی
- ۱۸۵ - مانت دستی رسانه قابل جابجایی
- ۱۸۷ - اتصال دائمی فایل سیستم ها در لینوکس
- ۱۸۷ - ساختار رکوردها در `/etc/fstab`
- ۱۸۸ - مثال `etc/fstab`
- ۱۸۸ - توضیح رکوردها
- ۱۸۹ - بررسی واحدهای Mount در `systemd`
- ۱۸۹ - نامگذاری فایل های Mount Unit
- ۱۸۹ - ساختار یک Mount Unit File
- ۱۹۰ - تست فایل Mount Unit
- ۱۹۱ - فعال کردن Mount Unit به صورت دائمی
- ۱۹۱ - مشاهده فایل سیستم های متصل
- ۱۹۱ - دستور `mountpoint`
- ۱۹۱ - دستور `blkid`
- ۱۹۲ - دستور `lsblk`
- ۱۹۲ - تغییر و مشاهده Label و UUID
- ۱۹۳ - موضوعات پیشرفته فایل سیستم ها
- ۱۹۳ - فایل سیستم های مبتنی بر حافظه
- ۱۹۳ - `proc filesystem` :
- ۱۹۴ - بررسی فایل سیستم Btrfs
- ۱۹۴ - ویژگی های کلیدی Btrfs
- ۱۹۴ - بررسی پشتیبانی Btrfs
- ۱۹۴ - فرمت و مانت Btrfs
- ۱۹۵ - مانت کردن فایل سیستم Btrfs

مشاهده وضعیت فایل سیستم Btrfs	۱۹۵ -
بررسی زیرحجم‌ها - Btrfs subvolume	۱۹۵ -
ساختار و مفهوم زیرحجم‌ها - subvolume	۱۹۵ -
کاربردهای زیرحجم‌ها - subvolume	۱۹۶ -
ایجاد یک زیرحجم	۱۹۶ -
مشاهده زیرحجم‌ها - subvolume	۱۹۶ -
دسترسی به زیرحجم‌ها - subvolume	۱۹۶ -
تبدیل فایل سیستم‌های دیگر به Btrfs	۱۹۷ -
سطح پیش‌فرض و مدیریت زیرحجم‌ها - subvolume	۱۹۷ -
مانت زیرحجم بدون دسترسی به حجم والد	۱۹۷ -
حذف زیرحجم	۱۹۸ -
بررسی Snapshots در Btrfs	۱۹۸ -
ایجاد Snapshot	۱۹۸ -
مدیریت Snapshots	۱۹۹ -
بررسی فایل سیستم‌های نوری (Optical Filesystems)	۱۹۹ -
انواع فایل سیستم‌های نوری	۱۹۹ -
دسترسی به رسانه نوری	۱۹۹ -
ایجاد فایل ISO با mkisofs یا genisoimage	۲۰۰ -
ایجاد ISO بوتیبل	۲۰۰ -
بررسی Swap Filesystems	۲۰۱ -
ایجاد و مدیریت Swap	۲۰۱ -
افزایش حافظه Swap در لینوکس	۲۰۳ -
بررسی Network-Based Filesystems	۲۰۴ -
انواع فایل سیستم‌های شبکه‌ای	۲۰۵ -
شبکه و ذخیره‌سازی	۲۰۵ -
درک فرآیند Auto-Mounting	۲۰۵ -
مدیریت پویا دستگاه‌ها	۲۰۵ -
مزایای auto-mounting :	۲۰۵ -
بررسی سرویس AutoFS	۲۰۵ -
مشکلات اتصال سیستم‌فایل‌های NFS با fstab	۲۰۵ -
راهکار AutoFS	۲۰۶ -
فایل‌های پیکربندی AutoFS	۲۰۶ -
بررسی Automount Units	۲۰۹ -
ساختار فایل پیکربندی Automount Unit	۲۰۹ -
بررسی Encrypted Filesystems یا سیستم‌فایل‌های رمزگذاری شده	۲۰۹ -

۲۰۹	مفهوم رمزگذاری (Encryption)
۲۱۰	فرآیند رمزگشایی (Decryption)
۲۱۰	اهمیت سیستم‌فایل‌های رمزگذاری شده
۲۱۰	dm-crypt
۲۱۱	eCryptfs
۲۱۱	Linux Unified Key Setup (LUKS)
۲۱۲	ساختار فایل پیکربندی Automount Unit
۲۱۲	ساختار کلی فایل Automount Unit
۲۱۳	گزینه‌های مهم در بخش [Automount]
۲۱۳	نحوه عملکرد Automount Unit
۲۱۳	مزایای استفاده از Automount Unit
۲۱۴	نگهداری از سیستم‌فایل‌ها
۲۱۵	تغییر و تنظیم سیستم‌فایل‌ها
۲۱۵	ابزارهای مدیریت ext2، ext3 و ext4
۲۱۷	ابزارهای مدیریت XFS
۲۱۷	ابزار xfs_admin
۲۱۷	ابزار xfs_fsr
۲۱۸	ابزار xfs_growfs
۲۱۸	ابزارهای مدیریت Btrfs
۲۱۸	ابزار btrfs balance
۲۱۹	ابزار btrfs-convert
۲۱۹	ابزار btrfstune
۲۱۹	ابزار btrfs property set
۲۲۰	مستندات (Man Pages)
۲۲۰	بررسی و ترمیم سیستم‌فایل‌ها
۲۲۰	ابزارهای بررسی و ترمیم در سیستم‌فایل XFS
۲۲۱	ترمیم با xfs_repair
۲۲۱	استخراج و بازسازی متادیتا با xfs_metadump و xfs_mdrestore
۲۲۱	نمایش اطلاعات سیستم‌فایل با xfs_info
۲۲۱	پشتیبان‌گیری و بازگردانی با xfsdump و xfsrestore
۲۲۲	ابزارهای بررسی و ترمیم در سیستم‌فایل Btrfs
۲۲۲	سیستم‌فایل‌های خراب
۲۲۲	استفاده از SMART
۲۲۳	سرویس smartd و فایل‌های پیکربندی
۲۲۳	کار با دستور smartctl
۲۲۴	لیست دستگاه‌های پشتیبانی شده
۲۲۴	اجرای تست با smartctl
۲۲۵	وضعیت سلامت کلی دستگاه

فصل ۵ - RAID و حجم‌های منطقی - ۲۲۸

پیکربندی RAID - ۲۲۸

درک سطوح مختلف RAID - ۲۲۸

RAID 0 – Striping - ۲۲۸

RAID 1 – Mirroring - ۲۲۹

RAID 10 – Striping + Mirroring (RAID 1+0) - ۲۲۹

RAID 2، ۳ و ۴ – ساختارهای منسوخ - ۲۲۹

RAID 5 – Striping with Distributed Parity - ۲۲۹

RAID 6 – Striping with Double Parity - ۲۳۰

نرم‌افزار و سخت‌افزار در RAID - ۲۳۱

پیاده‌سازی RAID - ۲۳۱

بررسی پشتیبانی سیستم از RAID - ۲۳۱

بررسی فایل /proc/mdstat - ۲۳۲

بارگذاری ماژول‌های RAID با modprobe - ۲۳۲

نصب ابزار مدیریت RAID (mdadm) - ۲۳۲

پارتیشن‌بندی دیسک‌ها - ۲۳۳

ایجاد پارتیشن روی یک دیسک (sde) : - ۲۳۳

ایجاد یک آرایه RAID - ۲۳۳

حالت‌های کاری mdadm - ۲۳۳

ایجاد آرایه با استفاده از mdadm - ۲۳۴

مفهوم Stripe و Chunk در RAID - ۲۳۶

قالب‌بندی (Format) و سوار کردن (Mount) آرایه RAID - ۲۳۸

ذخیره‌سازی پیکربندی RAID - ۲۳۸

ساخت فایل mdadm.conf - ۲۳۹

بررسی یک آرایه RAID - ۲۳۹

بررسی مفاهیم رابط‌های دیسک - ۲۴۰

PATA - ۲۴۰

SATA - ۲۴۰

ATAPI - ۲۴۰

SCSI - ۲۴۰

SAS - ۲۴۰

iSCSI - ۲۴۱

AHCI - ۲۴۱

Error! Bookmark not defined. Nvme

رابط‌های دیسک‌ها - ۲۴۲

تست دیسک‌ها - ۲۴۲

ابزار hdparm - ۲۴۲

- ۲۴۳ - تست عملکرد دیسک با `hdparm`
- ۲۴۴ - مثال‌های برای گزینه‌های `hdparm`
- ۲۴۴ - گزینه `i`
- ۲۴۴ - گزینه `I`
- ۲۴۴ - گزینه `t`
- ۲۴۴ - گزینه `T`
- ۲۴۵ - گزینه `W`
- ۲۴۵ - گزینه `X`
- ۲۴۵ - ابزار `sdparm`
- ۲۴۷ - ابزار `sysctl` برای تنظیم پارامترهای کرنل
- ۲۴۸ - بهینه‌سازی دسترسی به RAID با پارامترهای `sysctl`
- ۲۴۸ - استفاده از ابزار `smartctl`
- ۲۴۹ - استفاده از `smartd` daemon
- ۲۴۹ - استفاده از ابزار `nvme` برای SSD های `Nvme`
- ۲۴۹ - مثال‌های با `nvme`
- ۲۴۹ - فهرست دیسک‌ها و namespace ها
- ۲۵۰ - دستور `id-ctrl` — Identify Controller
- ۲۵۰ - دستور `id-ns` — Identify Namespace
- ۲۵۰ - ایجاد و حذف Namespace
- ۲۵۰ - اتصال و جدا کردن (attach / detach) Namespace
- ۲۵۱ - مشاهده گزارشات و وضعیت سلامت (Logs / SMART)
- ۲۵۱ - مدیریت Firmware
- ۲۵۱ - SSD و مدیریت عملکرد
- ۲۵۲ - دستور `fstrim` برای SSD ها
- ۲۵۲ - بررسی پشتیبانی SSD از TRIM
- ۲۵۲ - گزینه‌های دستور `fstrim`
- ۲۵۳ - اجرای دوره‌ای `fstrim`
- ۲۵۴ - پیاده‌سازی iSCSI در شبکه‌های ذخیره‌سازی
- ۲۵۴ - پروتکل‌های رایج SAN
- ۲۵۴ - مزایا و معایب iSCSI :
- ۲۵۵ - مفاهیم پایه iSCSI
- ۲۵۵ - iSCSI Qualified Name (IQN)
- ۲۵۵ - شناسایی جهانی دستگاه‌ها (WWID / WWN)
- ۲۵۶ - راه‌اندازی دیسک iSCSI در سرور Target
- ۲۵۹ - ایجاد iSCSI Target

ایجاد LUN	۲۶۱
راه اندازی دیسک iSCSI (روی Initiator کلاینت)	۲۶۲
نیازمندی ها و راه اندازی سرویس iscsid	۲۶۲
کشف دیسک (iSCSI Target Discovery)	۲۶۲
کشف Target با iscsiadm	۲۶۳
ورود به Target (Login)	۲۶۳
بررسی دیسک iSCSI متصل	۲۶۳
پارتیشن بندی و فرمت دیسک iSCSI	۲۶۴
مدیریت Logical Volume ها	۲۶۴
آشنایی با LVM	۲۶۵
اجزای اصلی LVM	۲۶۵
Physical Extent (PE)	۲۶۶
Logical Extent (LE)	۲۶۶
اهمیت و مزایای استفاده از LVM	۲۶۶
ایجاد حجم های منطقی (Creating Logical Volumes)	۲۶۷
آشنایی با دستورات مهم LVM	۲۶۷
تفاوت LVM1 و LVM2	۲۷۰
پنج گام اصلی برای ایجاد اولین LV	۲۷۰
ایجاد یک گروه حجم (Creating a VG)	۲۷۲
اهمیت انتخاب اندازه PE در VG	۲۷۲
نام گذاری گروه های حجم	۲۷۳
بررسی VG ساخته شده	۲۷۳
ایجاد یک حجم منطقی (Creating an LV)	۲۷۴
بررسی اطلاعات LV	۲۷۶
نمایش لیست تمام LV ها	۲۷۶
فرمت کردن و مونت کردن یک حجم منطقی (Formatting and Mounting an LV)	۲۷۶
مونت کردن LV	۲۷۷
نگهداری و مدیریت LV ها (Supporting Logical Volumes)	۲۷۷
افزایش ظرفیت گروه حجم و LV ها (Growing Your VGs and LVs)	۲۷۸
افزودن یک PV جدید به سیستم	۲۷۸
افزودن PV به VG	۲۷۸
افزایش اندازه LV با lvextend	۲۷۸
بررسی وضعیت LV افزایش یافته	۲۷۹
کاهش ظرفیت LV با lvreduce	۲۷۹
گسترش فضای Swap در LV	۲۸۰
ایجاد و نگهداری Snapshot در LV	۲۸۰

۲۸۰	مفهوم Snapshot در LVM
۲۸۰	فضای ذخیره سازی اسنپشات
۲۸۰	کاربردهای اصلی LV Snapshot
۲۸۱	ایجاد یک LV Snapshot
۲۸۱	بررسی وضعیت Snapshot
۲۸۲	مونت کردن Snapshot
۲۸۲	حذف Snapshot
۲۸۲	تغییر نام LV
۲۸۳	استفاده از فایل پیکربندی LVM
۲۸۴	کاربردهای عملی lvm.conf
۲۸۴	مشاهده تنظیمات lvm.conf
۲۸۴	افزودن و حذف Logical Volume ها

۲۸۵ - آشنایی با Device Mapper

۲۹۰ - فصل ۶

۲۹۰ - پیکربندی شبکه در لینوکس

۲۹۰	مبانی شبکه
۲۹۰	لایه فیزیکی (Physical Layer)
۲۹۰	شبکه سیم دار
۲۹۱	شبکه بی سیم
۲۹۱	لایه شبکه (Network Layer)
۲۹۱	آدرس IP
۲۹۱	IPv6
۲۹۱	روتر پیش فرض (Default Router)
۲۹۱	Netmask
۲۹۱	DNS و Hostname
۲۹۲	DHCP
۲۹۲	لایه انتقال (Transport Layer)
۲۹۲	لایه کاربردی (Application Layer)
۲۹۲	پیکربندی شبکه در لینوکس
۲۹۲	روش های پیکربندی
۲۹۳	فایل های پیکربندی شبکه
۲۹۴	جدول کامل پرتکل ها و پورت ها
۲۹۵	جدول تکمیلی پرتکل ها و پورت ها
۲۹۶	ابزارهای گرافیکی در مدیریت شبکه
۲۹۶	ابزارهای خط فرمان مدیریت شبکه
۲۹۶	شناسایی نام دستگاه شبکه
۲۹۷	گزینه های دستور ifconfig
۲۹۷	تنظیم شبکه های بی سیم

استفاده از DHCP	۲۹۸
ابزارهای جدید ip و iw	۲۹۸
عیب‌یابی پایه‌ای شبکه در لینوکس	۲۹۹
بررسی فایل‌های لاگ	۲۹۹
مشاهده حافظه نهان ARP	۲۹۹
ارسال بسته‌های آزمایشی	۳۰۰
آزمایش مسیرهای شبکه در لینوکس	۳۰۲
دستور traceroute	۳۰۲
دستور mtr (My Traceroute)	۳۰۲
آزمایش اتصال کلاینت و سرور	۳۰۳
یافتن اطلاعات میزبان در لینوکس	۳۰۵
دستور host	۳۰۵
بررسی چند آدرس IP	۳۰۵
یافتن نام میزبان از روی آدرس IP	۳۰۵
دستور dig	۳۰۶
بررسی رکورد A (IPv4)	۳۰۶
امنیت شبکه در لینوکس	۳۰۶
برنامه tcp_wrappers	۳۰۶
فایل‌های پیکربندی	۳۰۷
عیب‌یابی پیشرفته شبکه در لینوکس	۳۰۸
مشاهده اتصالات فعال شبکه	۳۰۸
دستور lsof	۳۰۸
دستور netstat	۳۰۹
مشاهده آمار شبکه	۳۰۹
دستور ss	۳۱۰
اسکن شبکه در لینوکس	۳۱۰
ضبط ترافیک شبکه با tcpdump	۳۱۱
نمونه ساده استفاده از tcpdump	۳۱۱
گزینه‌های کاربردی tcpdump	۳۱۱
ایمیل در لینوکس	۳۱۲
سه عملکرد اصلی سیستم ایمیل لینوکس	۳۱۳
عامل انتقال ایمیل (MTA)	۳۱۳
بسته‌های محبوب MTA در لینوکس	۳۱۴
Sendmail	۳۱۴

- ۳۱۴ -	Postfix
- ۳۱۵ -	Exim
- ۳۱۵ -	Mail Delivery Agent (MDA)
- ۳۱۵ -	روش mbox :
- ۳۱۵ -	روش Maildir :
- ۳۱۶ -	برنامه‌های MDA معروف
- ۳۱۶ -	Binmail
- ۳۱۶ -	Procmail
- ۳۱۶ -	Mail User Agent (MUA)
- ۳۱۶ -	برنامه‌های MUA (Mail User Agent)
- ۳۱۷ -	نمونه جلسه کاری با binmail :
- ۳۱۷ -	برنامه‌های MUA گرافیکی
- ۳۱۸ -	پروتکل‌های ایمیل
- ۳۱۸ -	Simple Mail Transfer Protocol (SMTP)
- ۳۱۸ -	دستورات پایه SMTP
- ۳۱۹ -	ارسال نمونه ایمیل با SMTP
- ۳۱۹ -	کدهای پاسخ SMTP
- ۳۲۰ -	Extended SMTP (ESMTP)
- ۳۲۱ -	پروتکل Post Office Protocol (POP)
- ۳۲۲ -	پروتکل Post Office Protocol (POP3)
- ۳۲۲ -	احراز هویت (Authentication) در POP3
- ۳۲۲ -	دستورات کلاینت POP3
- ۳۲۳ -	پروتکل Internet Message Access Protocol (IMAP)
- ۳۲۳ -	عملکرد IMAP
- ۳۲۴ -	احراز هویت IMAP
- ۳۲۵ -	Sieve زبان برنامه‌نویسی برای فیلتر ایمیل
- ۳۲۶ -	دستورات Sieve
- ۳۲۶ -	Action Commands دستورات عملیاتی
- ۳۲۶ -	Control Commands دستورات کنترلی
- ۳۲۶ -	Test Commands دستورات آزمون
- ۳۲۷ -	نمونه اسکریپت‌های Sieve
- ۳۲۸ -	مزایای Sieve نسبت به Procmail
- ۳۲۸ -	تحويل ایمیل از راه دور (Remote Email Delivery)
- ۳۲۸ -	استفاده از Courier
- ۳۲۸ -	فایل‌های پیکربندی Courier
- ۳۲۹ -	استفاده از Dovecot
- ۳۲۹ -	تنظیمات Dovecot در vacation
- ۳۲۹ -	مدیریت Dovecot با doveadm
- ۳۲۹ -	گزینه‌های مدیریت فرآیند اصلی Dovecot

- ۳۳۰ - گزینه‌های مدیریت کاربران و اتصالات
- ۳۳۰ - گزینه‌های مدیریت mailbox ها

فصل ۷ سیستم نام دامنه (DNS) و نرم‌افزار BIND - ۳۳۴

- ۳۳۴ - پیکربندی یک سرور DNS
- ۳۳۴ - درک DNS و BIND
- ۳۳۴ - نگاه کلی به DNS
- ۳۳۴ - ساختار پایگاه داده DNS
- ۳۳۵ - دامنه‌های سطح بالا (TLD)
- ۳۳۵ - فرآیند حل نام (Name Resolution)
- ۳۳۶ - حل نام از طریق Resolver
- ۳۳۷ - نگاه به BIND
- ۳۳۷ - جایگزین‌های BIND
- ۳۳۹ - نصب BIND در لینوکس
- ۳۳۹ - مقایسه سرویس‌ها و دایمون‌های BIND
- ۳۴۰ - بررسی فایل‌های BIND در لینوکس
- ۳۴۲ - Logging و سیستم ثبت گزارش‌ها
- ۳۴۲ - Zones یا مناطق DNS
- ۳۴۲ - Include Files و مدیریت فایل‌های پیکربندی
- ۳۴۲ - بررسی صحت فایل پیکربندی
- ۳۴۳ - فایل‌های پیکربندی زون پیش فرض
- ۳۴۳ - پیکربندی BIND روی لینوکس
- ۳۴۳ - سرور DNS فقط-کش (Caching-Only DNS Server)
- ۳۴۴ - نصب و ابزارهای BIND
- ۳۴۴ - نصب BIND
- ۳۴۴ - پیکربندی caching-only DNS
- ۳۴۵ - محدود کردن دسترسی به سرور caching-only
- ۳۴۶ - تنظیمات recursion
- ۳۴۶ - فایروال و دسترسی پورت ۵۳
- ۳۴۶ - راه‌اندازی سرویس BIND
- ۳۴۶ - تست بهبود زمان پاسخ DNS
- ۳۴۷ - پیکربندی سیستم‌های محلی برای استفاده از سرور caching-only
- ۳۴۷ - شروع، توقف و بارگذاری مجدد BIND
- ۳۴۷ - روش‌های سنتی (Traditional Methods)
- ۳۴۸ - استفاده از ابزار rndc
- ۳۴۸ - # rndc
- ۳۴۹ - پیکربندی لاگ‌گیری (Logging) در BIND
- ۳۵۰ - محل پیکربندی لاگ‌ها

بررسی Channel ها	۳۵۱
ایجاد و پیکربندی Channel های سفارشی	۳۵۱
دسته‌بندی‌ها (Categories)	۳۵۳
ایجاد و نگهداری زون‌های DNS (Creating and Maintaining DNS Zones)	۳۵۵
ساختار کلی یک زون DNS	۳۵۵
بررسی فایل‌های زون در BIND (Exploring BIND Zone Files)	۳۵۵
۱. فایل‌های پیکربندی زون (Zone Configuration Files)	۳۵۵
انواع زون‌ها در BIND	۳۵۶
انواع زون‌ها در BIND و مسیر فایل‌های آن‌ها	۳۵۸
زون Master	۳۵۸
زون Slave	۳۵۸
زون Forward	۳۵۹
زون Hint	۳۵۹
بررسی پایگاه داده‌های ناحیه (Zone Databases)	۳۶۰
مفهوم سرورهای اصلی و ثانویه	۳۶۰
دلایل رخ دادن Zone Transfer	۳۶۰
محل ذخیره فایل‌های Zone	۳۶۱
ساختار داده‌های Zone	۳۶۱
انواع Resource Record ها	۳۶۱
بررسی ناحیه‌های معکوس (Reverse Zones)	۳۶۳
نمونه پیکربندی Reverse Zone	۳۶۴
فایل پایگاه داده‌ی Reverse Zone	۳۶۴
تفاوت‌ها با Zone عادی:	۳۶۵
ساختار فایل زون معکوس	۳۶۵
ابزار بررسی صحت Zone Files	۳۶۶
مدیریت Zone ها در BIND	۳۶۶
تفویض اختیار (Zone Delegation) Zone	۳۶۷
چرا Zone Delegation ؟	۳۶۷
مراحل Zone Delegation :	۳۶۷
Glue Record چیست ؟	۳۶۷
ابزار dig	۳۶۸
ابزار nslookup	۳۶۹
ابزار rndc	۳۷۰
عیب‌یابی BIND با استفاده از rndc	۳۷۱
امن‌سازی سرور DNS	۳۷۳
راه‌اندازی امنیت پایه (Setting Up Basic Security)	۳۷۳
به‌روز نگه داشتن نرم‌افزار BIND	۳۷۳
مخفی کردن نسخه BIND	۳۷۳
استفاده از Views مختلف	۳۷۴

- ۳۷۵ - _____ DNS (Split DNS / Dual Horizon) تقسیم‌بندی سرور
- ۳۷۵ - _____ BIND به تنهایی اجرای
- ۳۷۵ - _____ root- به عنوان کاربر غیر- اجرای BIND
- ۳۷۵ - _____ (Zone Updates) کنترل به‌روزرسانی‌های زون
- ۳۷۵ - _____ (Zone Transfers) کنترل انتقال زون
- ۳۷۶ - _____ (Views and Split DNS) استفاده از Views و تقسیم‌بندی سرور
- ۳۷۸ - _____ BIND (Jailing BIND) محبوس کردن
- ۳۷۸ - _____ chroot jail مفهوم توضیح
- ۳۷۸ - _____ BIND برای chroot jail مراحل ایجاد
- ۳۷۹ - _____ تفاوت توزیع‌ها و بسته‌های آماده
- ۳۷۹ - _____ (Asymmetric Encryption and Hashing) کاوش در رمزنگاری نامتقارن و هشینگ
- ۳۷۹ - _____ (Asymmetric Encryption) رمزنگاری نامتقارن
- ۳۸۰ - _____ (Hashing) هشینگ
- ۳۸۰ - _____ DNSSEC استفاده از
- ۳۸۱ - _____ DNSSEC کاوش در
- ۳۸۱ - _____ بخش‌های اصلی پیام DNS :
- ۳۸۱ - _____ DNSSEC اصطلاحات مهم
- ۳۸۲ - _____ DNSSEC اعتبارسنجی داده‌های
- ۳۸۲ - _____ BIND در DNSSEC پیاده‌سازی
- ۳۸۳ - _____ DNSSEC تست عملکرد
- ۳۸۳ - _____ (Signing Your Zone) امضای زون
- ۳۸۴ - _____ TSIG (Transaction Signature) اتصال با
- ۳۸۶ - _____ DANE استفاده از
- ۳۸۶ - _____ DANE نحوه عملکرد
- ۳۸۷ - _____ DANE مزایای استفاده از
- ۳۸۹ - _____ DANE و TLSA روش کار
- ۳۸۹ - _____ DANE مزایا و اهمیت

فصل ۸ - استفاده از لینوکس به عنوان وب سرور - ۳۹۳ -

- ۳۹۳ - _____ وب سرور چیست؟
- ۳۹۳ - _____ HTTP
- ۳۹۳ - _____ HTML و سازماندهی داده‌ها
- ۳۹۴ - _____ نقش مرورگرها و وب سرورها
- ۳۹۴ - _____ فرآیند ساده مرورگر و وب سرور:
- ۳۹۴ - _____ HTTP استاندارد
- ۳۹۴ - _____ یک جلسه HTTP ساده:
- ۳۹۴ - _____ درخواست‌های کلاینت HTTP
- ۳۹۵ - _____ پاسخ‌های سرور (Server Responses)

۳۹۹ -	مکانیزم رمزنگاری در HTTPS
۳۹۹ -	فرآیند ارتباط در HTTPS
۴۰۰ -	وب سرورهای لینوکسی
۴۰۰ -	وب سرور Apache
۴۰۰ -	وب سرور Squid
۴۰۱ -	Nginx
۴۰۱ -	Load Balancer یا سرور تعادل بار
۴۰۲ -	وب سرور Apache
۴۰۳ -	نصب سرور Apache
۴۰۴ -	کنترل Apache
۴۰۴ -	ویژگی های پایه Apache
۴۰۴ -	ماژول های Apache
۴۰۵ -	پیکربندی Apache
۴۰۵ -	دستورهای رایج پیکربندی Apache
۴۰۷ -	لاگ های آپاچی (Apache Logs)
۴۰۷ -	میزبانی وب کاربران (User Web Hosting)
۴۰۸ -	میزبانی وب مجازی (Virtual Web Hosting)
۴۰۸ -	میزبانی مبتنی بر نام (Name-based)
۴۰۹ -	میزبانی مبتنی بر IP (IP-based)
۴۰۹ -	محدود کردن دسترسی (Access Restriction)
۴۱۰ -	میزبانی وب پویا (Hosting Dynamic Web Applications)
۴۱۱ -	رابط درگاه مشترک (CGI – Common Gateway Interface)
۴۱۱ -	ماژول های برنامه نویسی (Programming Modules)
۴۱۲ -	ایجاد یک وب سرور امن (Creating a Secure Web Server)
۴۱۲ -	نصب SSL
۴۱۳ -	ساخت کلید رمزنگاری (Encryption Key)
۴۱۳ -	ایجاد CSR (Certificate Signing Request)
۴۱۳ -	امضای CSR (Sign the CSR)
۴۱۴ -	نصب کلید و گواهی (Install the Key and Certificate)
۴۱۴ -	پیکربندی Apache برای استفاده از SSL (Configure Apache to Use SSL)
۴۱۵ -	استفاده از سرور پراکسی (Using a Proxy Server)
۴۱۵ -	نصب Squid (Installing Squid)
۴۱۶ -	پیکربندی Squid (Configuring Squid)
۴۱۶ -	وب کش (Web Cache)
۴۱۷ -	کنترل دسترسی وب (Web Access Control)
۴۱۷ -	احراز هویت کلاینت (Client Authentication)
۴۱۹ -	فصل ۹

به اشتراک گذاری فایل ها با استفاده از Samba، FTP و NFS ----- ۴۱۹ -

- ۴۱۹ - آشنایی با Samba
- ۴۱۹ - مفاهیم کلیدی در Samba
- ۴۲۰ - پیکربندی Samba
- ۴۲۰ - نصب Samba در لینوکس
- ۴۲۲ - بررسی دایرکتوری ها و فایل های Samba
- ۴۲۳ - بررسی ابزارهای Samba
- ۴۲۳ - پیکربندی یک اشتراک فایل در سرور Samba
- ۴۲۵ - بررسی دستورهای سراسری (Global Directives) در Samba
- ۴۲۶ - اهمیت تنظیم درست Workgroup در محیط های ترکیبی
- ۴۲۶ - بخش های غیر سراسری فایل smb.conf
- ۴۲۷ - پیکربندی اشتراک فایل Samba
- ۴۲۸ - بررسی صحت فایل smb.conf با testparm
- ۴۲۸ - افزودن حساب کاربری کلاینت
- ۴۲۹ - بررسی صحت حساب های کاربری
- ۴۲۹ - راه اندازی سرویس Samba
- ۴۲۹ - بررسی اشتراک های Samba
- ۴۳۰ - راه اندازی سرویس Samba در زمان بوت
- ۴۳۰ - پیکربندی فایروال برای Samba
- ۴۳۰ - پیکربندی کلاینت Samba
- ۴۳۲ - مدیریت امنیت و گروه ها
- ۴۳۲ - پیکربندی نقشه نام کاربری (Configuring a Username Map)
- ۴۳۳ - پیکربندی سطح امنیت (Configuring Security Levels)
- ۴۳۳ - پیکربندی سامبا (Samba) به عنوان عضو دامنه Active Directory
- ۴۳۶ - پیکربندی اشتراک چاپگر در سرور سامبا
- ۴۳۸ - راه اندازی مجدد سرویس سامبا
- ۴۳۸ - بررسی اشتراک چاپگر با smbclient
- ۴۳۹ - ارسال کار چاپ از کلاینت
- ۴۳۹ - مشاهده صف چاپ در سرور
- ۴۴۱ - درک (Understanding NFS) NFS
- ۴۴۲ - نحوه عملکرد NFS
- ۴۴۲ - NFSv2
- ۴۴۲ - NFSv3
- ۴۴۳ - NFSv4
- ۴۴۳ - تاکید بر LPIC-2 بر NFSv3
- ۴۴۳ - Share یا Export
- ۴۴۳ - الزامات NFS در لینوکس
- ۴۴۴ - منابع مستندات NFS:
- ۴۴۴ - بررسی ابزارهای (Exploring NFS Utilities) NFS

۴۴۵	ابزار rpcinfo
۴۴۵	ابزار exportfs
۴۴۶	غیرفعال کردن NFSv4 (Disabling NFSv4)
۴۴۶	پیکربندی یک NFS Export موقت (Configuring a Temporary NFS Export)
۴۴۷	پیکربندی NFS Export دائمی (Permanent NFS Export)
۴۴۸	امنیت NFS (Securing NFS)
۴۴۹	ایمن‌سازی NFSv3 (Securing NFSv3)
۴۴۹	نگاهی به سرورهای FTP
۴۴۹	درک اتصال‌های Active و Passive
۴۵۰	نگاهی به سرورها و کلاینت‌های FTP
۴۵۰	Very Secure FTP (vsftpd)
۴۵۰	Pure-FTPd (pure-ftpd)
۴۵۰	ProFTPD
۴۵۱	کلاینت‌های FTP
۴۵۱	مرورگرهای وب
۴۵۱	برنامه‌های گرافیکی (GUI)
۴۵۱	ابزارهای خط فرمان
۴۵۱	پیکربندی vsftpd
۴۵۲	دسترسی با نام کاربری و رمز عبور
۴۵۴	کنترل دسترسی با TCP Wrappers
۴۵۵	دسترسی ناشناس برای دانلود
۴۵۶	دسترسی ناشناس برای آپلود فایل‌ها
۴۵۷	پیکربندی Pure-FTPd
۴۵۷	بررسی گزینه‌های خط فرمان (Command-Line Options)
۴۵۸	کنترل از طریق فایل پیکربندی
۴۵۹	دسترسی با نام کاربری یا ناشناس
۴۶۰	تست آپلود و دانلود ناشناس در Ubuntu
۴۶۰	اختصاص آدرس‌های شبکه
۴۶۰	آدرس‌های IP ثابت
۴۶۱	آدرس‌های IP پویا
۴۶۱	استاندارد DHCP
۴۶۱	گزینه‌های DHCP
۴۶۲	پیاده‌سازی DHCP در لینوکس
۴۶۳	نرم‌افزار DHCP در لینوکس
۴۶۳	نصب سرور DHCP در لینوکس

۴۶۴	پیکربندی سرور DHCP
۴۶۴	گزینه‌های سراسری (Global Options)
۴۶۴	تعریف محدوده‌های زیرشبکه (Setting Subnet Ranges)
۴۶۵	تعریف چند زیرشبکه
۴۶۵	پیگیری آدرس‌های IP اختصاص داده شده
۴۶۵	میزبان‌های ثابت (Static Hosts)
۴۶۷	میزبان‌های BOOTP
۴۶۷	رله DHCP (DHCP Relaying)
۴۶۸	فایل‌های لاگ DHCP (DHCP Log Files)
۴۶۸	پیکربندی کلاینت‌ها (Configuring Clients)
۴۶۹	سرویس احراز هویت (Authentication Service)
۴۶۹	مبانی PAM
۴۷۰	مثال مازول‌های PAM
۴۷۰	پیکربندی PAM (Configuring PAM)
۴۷۳	دایرکتوری‌های شبکه (Network Directories)
۴۷۳	اصول LDAP (LDAP Basics)
۴۷۳	ساختار پایگاه داده LDAP
۴۷۴	اجزای پایگاه داده LDAP
۴۷۴	نامگذاری اشیاء در LDAP
۴۷۴	فرمت تبادل داده LDAP (LDIF)
۴۷۵	ذخیره‌سازی پایگاه داده LDAP
۴۷۵	Stand-alone Mode
۴۷۵	Replicated Mode
۴۷۵	Distributed Mode
۴۷۵	سرور OpenLDAP
۴۷۷	نصب OpenLDAP
۴۷۷	برنامه‌های سرور OpenLDAP
۴۷۷	برنامه slapd
۴۷۸	برنامه slurpd
۴۷۹	طراحی اسکیمای (Schema) دایرکتوری
۴۷۹	پیاده‌سازی اسکیمای دایرکتوری (Implementing a Directory Schema)
۴۷۹	روش slapd.conf
۴۸۰	دسترسی و کنترل‌ها
۴۸۰	ابزارهای کمکی OpenLDAP
۴۸۰	افزودن اشیاء با slapadd

۴۸۱	روش slapd-config
۴۸۱	پیاده‌سازی کلاینت‌های LDAP
۴۸۳	تفاوت slapadd و ldapadd :
۴۸۳	ابزار اصلی کلاینت ldapsearch
۴۸۴	امنیت در لینوکس
۴۸۵	امنیت شبکه سرور
۴۸۵	اسکن پورت (Port Scanning)
۴۸۶	Telnet
۴۸۶	netstat
۴۸۷	Netcat (nc)
۴۸۷	nmap
۴۸۷	OpenVAS
۴۸۸	سیستم‌های تشخیص نفوذ (IDS – Intrusion Detection Systems)
۴۸۸	fail2ban
۴۸۹	Snort
۴۸۹	حالت‌های Snort
۴۹۰	امنیت شبکه خارجی (External Network Security)
۴۹۰	آدرس‌های شبکه خصوصی و NAT
۴۹۱	فایروال‌ها (Firewalls)
۴۹۲	استفاده از iptables
۴۹۴	مسیریابی در لینوکس (Routing in Linux)
۴۹۵	اتصال امن به سرور (Connecting Securely to a Server)
۴۹۵	OpenSSH
۴۹۵	فایل‌های OpenSSH
۴۹۶	پیگر بندی OpenSSH
۴۹۶	استفاده از OpenSSH
۴۹۷	انتقال فایل با OpenSSH
۴۹۷	تونل زنی X11 با OpenSSH
۴۹۷	استفاده از Client Certificates گواهی‌های کلاینت
۴۹۸	OpenVPN
۴۹۸	نصب و پیگر بندی OpenVPN
۴۹۹	پیگر بندی اتصال VPN
۵۰۰	استفاده از OpenVPN
۵۰۰	منابع امنیتی (Security Resources)

- 500 - ----- US-CERT

- 501 - ----- SANS Institute

مقدمه

پس از موفقیت و استقبال گسترده از کتاب مرجع دستورات لینوکس تحت عنوان **۱۰۰۱ دستور لینوکس**، این بار با افتخار کتابی جامع و کاربردی با عنوان **کتاب مرجع کامل مدرک بین‌المللی LPIC-2** را در اختیار شما قرار می‌دهیم. این کتاب نه تنها آموزش گام‌به‌گام و کامل مباحث اصلی لینوکس را ارائه می‌دهد، بلکه با بهره‌گیری از منابع بین‌المللی معتبر و تجربه‌های عملی، مطالعه‌ای منسجم، عمیق و کاربردی را برای شما فراهم می‌آورد.

کتاب حاضر تمامی مفاهیم و سرفصل‌های مورد نیاز برای موفقیت در آزمون‌های **LPIC-2** را پوشش می‌دهد، به گونه‌ای که نیاز به مراجعه به کتاب‌ها و منابع دیگر به حداقل می‌رسد. این کتاب شامل محتوای جامع و به‌روز شده از منابع معتبر زیر است:

• **LPIC-2 Objectives V4.5 – Linux Professional Institute**

• **LPIC-2: Study Guide: Exam 201 and Exam 202 (Sample, 2nd Edition)**

• **LPIC-2 Cert Guide: (201-450 and 202-450 exams) (Pearson IT Certification)**

• **CompTIA LPIC-2 201, and LPIC-2 202 (Sample)**

• **GNU/Linux Advanced Administration**

با مطالعه این کتاب، شما نه تنها برای آزمون‌های **LPIC-2** آماده خواهید شد، بلکه دانش تئوری و مهارت عملی شما در لینوکس به سطحی حرفه‌ای و قابل اعتماد ارتقا می‌یابد. این مجموعه می‌تواند هم به عنوان یک راهنمای آموزشی گام‌به‌گام و هم به عنوان یک مرجع سریع و کاربردی در کنار شما باشد و نیازهای شما را در محیط‌های آموزشی، کاری و حرفه‌ای به طور کامل پوشش دهد.

مخاطبین هدف این دوره چه کسانی هستند؟

این دوره به گونه‌ای طراحی شده است که طیف وسیعی از علاقه‌مندان و متخصصان فعال در حوزه فناوری اطلاعات را پوشش دهد. محتوای آموزشی از مبانی پایه شروع شده و تا موضوعات پیشرفته و **کاربردی** در محیط‌های enterprise ادامه می‌یابد، بنابراین برای افراد تازه‌کار و همچنین حرفه‌ای‌هایی که قصد تثبیت دانش و پر کردن خلاءهای علمی خود را دارند، کاملاً مناسب است.

- تازه‌واردان به دنیای لینوکس: افرادی که هیچ پیش‌زمینه‌ای ندارند و می‌خواهند یادگیری لینوکس را از پایه‌ترین مفاهیم به صورت اصولی و **کاربردی** آغاز کنند.
- جامعه تو سعه‌دهندگان و مهند سین نرم‌افزار: برنامه‌نوی سان، تو سعه‌دهندگان وب و مهند سین نرم‌افزار که برای توسعه، استقرار (Deploy) و **عیب‌یابی** (Debug) برنامه‌های خود نیاز به تسلط بر محیط سرورهای لینوکس و خط فرمان (Terminal) دارند.
- متخصصان DevOps و ابر (Cloud): کار شناسان DevOps، مهند سین (SRE) Site Reliability و متخصصان Cloud که اسکریپت‌نویسی، اتوماسیون، مدیریت زیر ساخت به‌عنوان کد (IaC) و کار با سرویس‌های ابری مانند AWS، Azure و GCP در هسته مرکزی فعالیت آن‌ها قرار دارد.
- متخصصان و مدیران شبکه: مدیران شبکه، مهند سین زیر ساخت IT، کار شناسان NOC و آنالیزورهای شبکه که برای مدیریت روترها، سوئیچ‌ها، فایروال‌ها و سرویس‌های تحت شبکه (مانند DNS, DHCP, Proxy) به لینوکس به‌عنوان یک ابزار حیاتی نیاز دارند.
- حرفه‌ای‌های امنیت سایبری: کارشناسان تست نفوذ (Penetration Testers)، هک‌های قانونمند (Ethical Hackers)، متخصصان امنیت اطلاعات (SOC Analysts) و پاسخ به حوادث (Incident Responders) که برای تحلیل تهدیدات، بررسی لاگ‌ها و استفاده از ابزارهای امنیتی، نیازمند تسلط عمیق بر سیستم عامل لینوکس هستند.
- تیم‌های پشتیبانی فنی و عملیات: تکن‌سین‌های پشتیبانی (Help Desk)، متخصصان صین پشتیبانی سرور و اپراتورهای دیتاسنتر که وظیفه **عیب‌یابی**، مانیتورینگ سلامت سرویس‌ها و انجام عملیات اولیه را بر عهده دارند.
- مشاوران و مدیران فناوری اطلاعات: مشاوران IT و شبکه، مدیران فنی و مدیران ارشد فناوری اطلاعات (CTOs) که برای تصمیم‌گیری‌های استراتژیک، انتخاب پلتفرم‌های مناسب و درک قابلیت‌های سیستم‌عامل لینوکس نیاز به دانش کافی دارند.
- دانشجویان و پژوهشگران: دانشجویان مقاطع مختلف در رشته‌های کامپیوتر، فناوری اطلاعات، مهندسی نرم‌افزار و شبکه که می‌خواهند دانش آکادمیک خود را با مهارت‌های عملی و مورد نیاز بازار کار تکمیل کنند.

فصل یکم

فصل ۱

تاریخچه سیستم عامل لینوکس

برگرفته از خانواده سیستم‌عامل‌های یونیکس است. برخلاف بسیاری از سیستم‌عامل‌های انحصاری، لینوکس تحت مالکیت یا نظارت یک شرکت یا سازمان واحد نیست؛ بلکه حاصل همکاری گسترده میان شرکت‌های تجاری، سازمان‌های غیرانتفاعی و هزاران توسعه‌دهنده مستقل در سراسر جهان است. این همکاری جمعی باعث شده لینوکس به یکی از انعطاف‌پذیرترین و قابل‌اعتمادترین سیستم‌عامل‌های موجود تبدیل شود.

تاریخچه سیستم عامل یونیکس

سرآغاز داستان لینوکس را باید در یونیکس (UNIX) جستجو کرد. نخستین نسخه از یونیکس در سال ۱۹۶۹ توسط کن تامپسون، دنیس ریچی و همکارانشان در آزمایشگاه‌های بل (Bell Labs) توسعه یافت. (جالب است بدانید که همان سال، لینوس توروالدز، خالق لینوکس، به دنیا آمد. در ابتدا، یونیکس به زبان اسمبلی برای رایانه‌های مینی (Mini Computers) نوشته شد. اما در اوایل دهه ۷۰، دنیس ریچی زبان برنامه‌نویسی C را معرفی کرد که یک زبان سطح بالا و قابل‌حمل بود. این تحول باعث شد که بخش‌های زیادی از کرنل یونیکس به زبان C بازنویسی شوند و این امر قابلیت انتقال‌پذیری (Portability) سیستم‌عامل را به‌طور چشمگیری افزایش داد.

از یونیکس تا لینوکس — روایتی از تولد یک جنبش نرم‌افزاری آزاد

آغاز ماجرا: تولد یونیکس ۱۹۶۹

در اواخر دهه ۱۹۶۰، در آزمایشگاه‌های بل (Bell Labs)، سه برنامه‌نویس برجسته به نام‌های کن تامپسون، دنیس ریچی و داگلاس مک‌ایلروی در حال کار بر روی پروژه‌های بزرگ و پیچیده رایانه‌ای بودند. نتیجه تلاش‌های آنان در سال ۱۹۶۹ به تولد سیستم‌عامل یونیکس (UNIX) انجامید. جالب است بدانید که همان سال، در سوی دیگری از جهان، لینوس توروالدز، کسی که بعدها لینوکس را خلق کرد، به دنیا آمد. نسخه اولیه یونیکس به زبان اسمبلی نوشته شد و برای اجرا روی رایانه‌های PDP-11 طراحی شده بود. این سیستم‌عامل به دلیل سادگی، کارایی و قابلیت چندکاربره بودن، خیلی زود توجه محققان و دانشگاه‌ها را به خود جلب کرد.

زبان C و تحولی بزرگ ۱۹۷۳

در سال ۱۹۷۳، دنیس ریچی زبان C را طراحی کرد. این زبان سطح بالا به اندازه کافی قدرتمند بود که بتواند هسته سیستم‌عامل را پیاده‌سازی کند و در عین حال، قابلیت انتقال بین سخت‌افزارهای مختلف را فراهم آورد. نسخه چهارم یونیکس، نخستین نسخه‌ای بود که تمام هسته آن به زبان C نوشته شد؛ اتفاقی که انقلاب بزرگی در دنیای نرم‌افزار ایجاد کرد.

گسترش در دانشگاه‌ها و شکل‌گیری شاخه‌ها

کد منبع یونیکس همراه با مستندات آن به دانشگاه‌ها ارسال شد. این اقدام باعث شد هزاران دانشجو و محقق بتوانند ساختار و کد سیستم‌عامل را مطالعه، تغییر و بهبود دهند. تا اواخر دهه ۷۰، یونیکس در بیش از ۱۲۰ دانشگاه و ۵۰۰ مرکز تحقیقاتی در سراسر جهان نصب شده بود. در این زمان، یونیکس به دو شاخه اصلی تقسیم شد:

۱. BSD (Berkeley Software Distribution) توسعه‌یافته در دانشگاه کالیفرنیا، برکلی، با تمرکز بر

قابلیت‌های شبکه و ابزارهای پیشرفته.

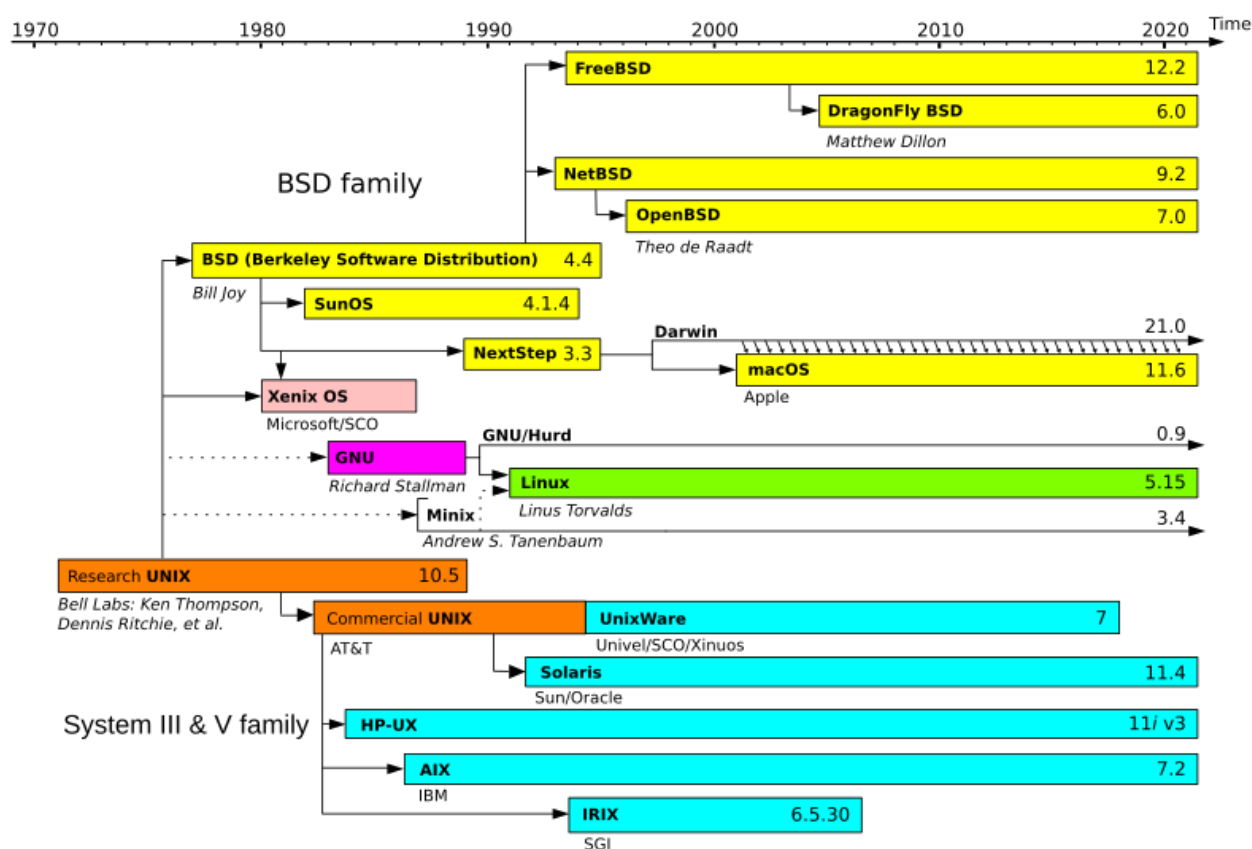
۲. System V توسعه یافته توسط AT&T، با ساختار سازمانی و استانداردسازی گسترده.

ظهور مینیکس — پلی میان آموزش و عمل ۱۹۸۷

دهه ۸۰ با رشد سریع رایانه های شخصی همراه بود، اما بیشتر سیستم عامل ها یا انحصاری و گران بودند (مثل محصولات مایکروسافت) یا پیچیده و گران قیمت (مثل یونیکس). در این میان، اندرو تاننباوم، استاد علوم رایانه در هلند، سیستم عامل مینیکس (MINIX) را طراحی کرد. مینیکس یک نسخه کوچک و آموزشی از یونیکس بود که همراه با کتاب تاننباوم منتشر می شد. این کتاب حاوی توضیحات کامل و حدود ۱۲ هزار خط کد به زبان C و اسمبلی بود و به دانشجویان اجازه می داد منطق داخلی یک سیستم عامل واقعی را درک کنند. مینیکس متن باز بود (هرچند با محدودیت های مجوزی اولیه) و همین باعث شد افراد زیادی از جمله لینوس توروالدز با آن آشنا شوند.

ریچارد استالمن و پروژه گنو ۱۹۸۳

در سال ۱۹۸۳، ریچارد استالمن، برنامه نویس آمریکایی و فعال حوزه آزادی نرم افزار، پروژه ای را به نام GNU آغاز کرد. هدف او ساخت یک سیستم عامل کاملاً آزاد و سازگار با یونیکس بود. پروژه گنو ابزارها، کامپایلرها، ویرایشگر متن و بسیاری از اجزای مهم سیستم عامل را تولید کرد، اما هسته (Kernel) آن — به نام Hurd — کامل نشد. این کمبود بعدها با هسته لینوکس جبران شد.



فصل دوم

فرایند Boot (بوت) و Startup (راه اندازی)

دقیقاً مانند راهنمای مطالعه LPIC-1، این کتاب شامل توضیحات مفصل در مورد تمامی اهداف آزمون LPIC-2 است. هدف این کتاب کمک به شما برای موفقیت در هر دو آزمون LPIC-2، یعنی ۲۰۱ و ۲۰۲ است. این آزمون‌ها موضوعات پیشرفته‌تری نسبت به آزمون LPIC-1 را پوشش می‌دهند؛ از جمله هسته لینوکس، راه‌اندازی سیستم، فایل سیستم‌ها، عملیات شبکه، سرورهای DNS، سرورهای وب، سرورهای فایل، سرورهای ایمیل، مدیریت کلاینت‌های شبکه و امنیت. این کتاب شما را گام‌به‌گام در تمامی این مباحث همراهی می‌کند تا برای پرسش‌های آزمون LPIC-2 آماده شوید.

هدف برنامه LPIC-2 از سوی *Linux Professional Institute (LPI)* این است که سطح دانش لازم برای مدیریت شبکه‌های کوچک تا متوسط ترکیبی (ترکیب شده از میکروسافت و لینوکس) را تعریف کند، با تمرکز ویژه بر روی سیستم‌عامل لینوکس. این برنامه برای متخصصانی طراحی شده که می‌خواهند دانش خود را که در سطح LPIC-1 کسب کرده‌اند، توسعه دهند و به مرحله‌ای بالاتر برسانند.

انتظار می‌رود داوطلبان پیش از ورود به آزمون LPIC-2، آزمون **Linux Essentials** (اختیاری) و آزمون **LPIC-1** یا آزمون معادل آن یعنی **CompTIA Linux+** را گذرانده باشند.

فردی که در آزمون LPIC-2 موفق شود، باید در کمترین حالت دارای دانش و تجربه در زمینه‌های زیر باشد:

- مدیریت چندین سرور لینوکس
- ارائه مشاوره به مدیران در زمینه مکانیزه‌سازی و خرید تجهیزات
- برنامه‌ریزی و مدیریت یک شبکه کوچک ترکیبی، شامل:
 - **LAN Server** سرور شبکه محلی
 - مدیریت کلاینت‌ها
 - سرویس **DHCP** اختصاص‌دهی پویا به IP ها
 - سرویس **DNS** سامانه نام دامنه
 - **NFS** اشتراک فایل در لینوکس
 - **Samba** ارتباط لینوکس با ویندوز برای اشتراک فایل و پرینتر
 - **Internet Gateway** دروازه اینترنت
 - **Firewall** دیوار آتش
 - سرویس‌های **Mail** ایمیل
 - **OpenSSH** ارتباط امن و مدیریت راه دور
 - شبکه خصوصی مجازی **VPN**
 - **Web Cache/Proxy** کش وب و پراکسی برای مدیریت و کنترل دسترسی
 - **Internet Server** سرور اینترنت
 - **FTP Server** انتقال فایل
 - **Web Server** راه‌اندازی و مدیریت وبسایت‌ها
 - وب سرور با **Reverse Proxy** جهت افزایش امنیت و کارایی

ساختار **ESP** از سیستم فایل قدیمی **FAT – File Allocation Table** (جدول تخصیص فایل) میکروسافت استفاده می کند تا بوت لودرها را ذخیره کند. در سیستم های لینوکسی، این پارتیشن معمولاً در مسیر **boot/efi** مونت (mount) می شود و فایل های بوت لودر معمولاً با پسوند **efi** ذخیره می گردند.

UEFI Boot Manager مدیر بوت UEFI

میان افزار **UEFI** دارای یک بوت لودر کوچک داخلی است که اغلب به آن **Boot Manager** (مدیر بوت) گفته می شود. این ابزار داخلی به شما اجازه می دهد که تعیین کنید کدام فایل بوت لودر باید اجرا شود. البته، همه توزیع های لینوکس از **UEFI Firmware** پشتیبانی نمی کنند. بنابراین اگر از سیستمی مبتنی بر **UEFI** استفاده می کنید، باید مطمئن شوید توزیع لینوکسی انتخابی شما آن را پشتیبانی می کند. در **UEFI** لازم است که هر فایل بوت لودری را که می خواهید در منوی بوت نمایش داده شود، در **Boot Manager Interface Menu** (منوی رابط مدیر بوت) ثبت کنید. برای این کار، در لینوکس برنامه ای به نام **efibootmgr** وجود دارد که به شما امکان می دهد ورودی های بوت را ایجاد یا حذف کنید و ترتیب بوت را تغییر دهید. رابط **UEFI** همچنین شامل یک محیط **Shell** (شل یا محیط دستوری داخلی) است که به شما امکان می دهد دستورات لازم برای تغییر تنظیمات بوت را وارد کنید یا مشخص کنید که هنگام هر بار بوت، کدام بوت لودر اجرا شود.

پشتیبانی از SSD و Nvme

یکی دیگر از فناوری های محبوب و جدید که معمولاً در سیستم های مبتنی بر **UEFI** یافت می شود، جایگزینی **Legacy Hard Drives** (هارد های سنتی) با **SSD – Solid State Drive** (درایو حالت جامد) است. سخت افزارهای **SSD** از کنترلرهای قدیمی هارد استفاده نمی کنند، بلکه از استاندارد **Nvme – Non-Volatile Memory Express** (رابط حافظه غیر فرار پرسرعت) بهره می برند.

Linux Kernel (هسته لینوکس) از نسخه ۳٫۳ به بعد از سیستم های **Nvme** پشتیبانی می کند و فرایند بوت **UEFI** نیز در هنگام استفاده از سخت افزار **SSD** در سیستم های لینوکسی بدون مشکل کار می کند.

Linux Bootloaders بوت لودرهای لینوکس

برنامه **Bootloader** (بوت لودر) نقش پل ارتباطی میان **Firmware** (میان افزار سیستم) و **Linux Kernel** (هسته لینوکس) را ایفا می کند. در لینوکس چندین انتخاب برای بوت لودر وجود دارد. پرکاربردترین آن ها که احتمالاً با آن ها روبه رو خواهید شد عبارتند از:

LILO – Linux Loader (بارگذار لینوکس)

GRUB Legacy – Grand Unified Bootloader Legacy (بوت لودر قدیمی گراب)

GRUB2 – Grand Unified Bootloader v2 (نسخه دوم بوت لودر گراب)

LILO – Linux Loader

در نسخه های اولیه لینوکس، تنها بوت لودر موجود **LILO** بود. این بوت لودر قابلیت های محدودی داشت، اما هدف اصلی خود را برآورده می کرد: یعنی بارگذاری **Linux Kernel** از طریق راه اندازی **BIOS**

LILO (Linux Loader) یکی از اولین بوت لودرهای سیستم عامل لینوکس بود که در دهه ۱۹۹۰ میلادی برای بارگذاری هسته لینوکس طراحی شد. در آن زمان، سیستم های مبتنی بر **BIOS (Basic Input/Output System)** رایج بودند و **LILO** به عنوان ابزاری ساده و کارآمد برای بارگذاری هسته لینوکس از دیسک سخت به حافظه عمل می کرد. در ابتدا، **LILO** به عنوان بوت لودر پیش فرض در بسیاری از توزیع های لینوکس مورد استفاده قرار می گرفت. فایل پیکربندی آن معمولاً در مسیر **etc/lilo.conf** قرار داشت و کاربران می توانستند سیستم عامل ها و هسته های مختلف را در این فایل تعریف کنند. **LILO** به صورت مستقیم از طریق **BIOS** سیستم بارگذاری می شد و نیازی به پشتیبانی از ویژگی های پیشرفته ای مانند **UEFI (Unified Extensible Firmware Interface)** نداشت.

تعامل با GRUB2

بوت‌لودر GRUB2 هنگام راه‌اندازی سیستم، یک **Boot Menu منوی بوت** (نمایش می‌دهد که بسیار شبیه به نسخه‌ی قدیمی‌تر یعنی GRUB Legacy است. از طریق این منو می‌توانید سیستم‌عامل موردنظر خود را برای بوت انتخاب کرده یا تنظیمات بوت را تغییر دهید.

کلیدهای کاربردی در منوی بوت GRUB2

• کلیدهای جهت‌نما (Arrow Keys) :

برای جابه‌جایی بین گزینه‌های مختلف بوت از کلیدهای بالا و پایین استفاده کنید.

• کلید E :

برای **ویرایش موقت تنظیمات بوت** (مانند تغییر پارامترهای کرنل یا مسیر بوت) روی گزینه‌ی انتخاب‌شده فشار دهید.

پس از انجام تغییرات، با فشردن کلید **Ctrl + X** یا **F10** سیستم را با تنظیمات جدید بوت کنید.

• کلید C :

وارد **GRUB Command Line** خط فرمان (GRUB2) می‌شوید.

در این حالت می‌توانید دستورات تعاملی مانند **ls** (برای فهرست دیسک‌ها و پارتیشن‌ها) یا **set** (برای مشاهده متغیرهای بوت) را وارد کنید.

نکته در مورد رابط‌های گرافیکی

در برخی توزیع‌های لینوکس با رابط گرافیکی (مثل **Ubuntu**، منوی بوت GRUB2 معمولاً **پنهان** است تا فرآیند بوت سریع‌تر و زیباتر باشد. اگر بخواهید این منو را ببینید یا به تنظیمات آن دسترسی پیدا کنید، کافی است **در لحظه‌ی روشن شدن سیستم، کلید Shift را نگه دارید** در سیستم‌های (BIOS یا کلید **Esc** را چند بار فشار دهید در سیستم‌های EFI / UEFI).

بوت‌لودرهای جایگزین

هرچند **GRUB Legacy** و **GRUB2** محبوب‌ترین بوت‌لودرهای لینوکس هستند، اما در برخی توزیع‌ها یا شرایط خاص ممکن است با بوت‌لودرهای دیگر نیز مواجه شوید.

Systemd-boot

بوت‌لودر **Systemd-boot** (سیستم‌دی-بوت) به‌ویژه در توزیع‌هایی که از روش **systemd init** راه‌اندازی (systemd) استفاده می‌کنند، محبوبیت پیدا کرده است. این بوت‌لودر یک **Boot Menu** ایجاد می‌کند و می‌تواند هر **EFI Boot Image** ایمج بوت EFI را بارگذاری کند.

U-Boot

بوت‌لودر **U-Boot** (یوبوت) توانایی بوت از هر نوع **Disk** (دیسک) و بارگذاری هر نوع **Boot Image** (ایمج بوت) را دارد.

Syslinux Project پروژه سیسلینوکس

پروژه **Syslinux** شامل پنج بوت‌لودر جداگانه است که هرکدام کاربرد خاصی دارند:

- **SYSLINUX** بوت‌لودر مخصوص سیستم‌هایی که از **Microsoft FAT Filesystem** فایل سیستم FAT مایکروسافت استفاده می‌کنند. این بوت‌لودر معمولاً برای بوت از **USB Memory Stick** فلش (USB) رایج است.
- **EXTLINUX** یک مینی بوت‌لودر برای بوت از فایل سیستم‌های **ext2**، **ext3**، **ext4** یا **btrfs**.

برنامه **systemctl** با استفاده از **دستورات (Commands)** مختلف مشخص می‌کند که چه عملیاتی روی یک واحد انجام شود. این دستورات می‌توانند وضعیت سرویس‌ها را بررسی کنند، سرویس‌ها را شروع یا متوقف کنند، فایل‌های پیکربندی را بارگذاری مجدد کنند و حتی سطح اجرای سیستم یا Target فعلی را تغییر دهند. جدول زیر مهم‌ترین دستورات **systemctl** و کاربرد آن‌ها را نشان می‌دهد:

دستور	توضیح
<i>list-units</i>	نمایش وضعیت کنونی تمام واحدهای پیکربندی شده و فعال سیستم
<i>default</i>	تغییر به Target پیش‌فرض سیستم، یعنی سطحی که سیستم در هنگام بوت به آن می‌رود
<i>isolate</i>	اجرای یک واحد مشخص و توقف تمام واحدهای دیگر؛ برای تغییر کامل سطح اجرایی سیستم
<i>start name</i>	شروع واحد مشخص شده، مانند یک سرویس یا Target
<i>stop name</i>	متوقف کردن واحد مشخص شده
<i>reload name</i>	بارگذاری مجدد فایل پیکربندی واحد مشخص شده بدون توقف کامل آن
<i>restart name</i>	متوقف و دوباره راه‌اندازی کردن واحد مشخص شده؛ برای اعمال تغییرات اساسی یا رفع مشکلات
<i>status name</i>	نمایش وضعیت واحد مشخص شده؛ شامل PID، وضعیت فعال یا غیرفعال بودن و مسیر فایل پیکربندی
<i>enable name</i>	پیکربندی واحد برای شروع خودکار هنگام بوت بعدی
<i>disable name</i>	پیکربندی واحد برای عدم شروع خودکار هنگام بوت بعدی

یک جدول با مثال برای هر دستور **systemd**

مثال	توضیح	دستور
systemctl list-units	نمایش وضعیت کنونی تمام واحدهای پیکربندی شده و فعال سیستم	<i>list-units</i>
systemctl get-default systemctl set-default graphical.target	تغییر به Target پیش‌فرض سیستم، یعنی سطحی که سیستم در هنگام بوت به آن می‌رود	<i>default</i>
systemctl isolate multi-user.target	اجرای یک واحد مشخص و توقف تمام واحدهای دیگر؛ برای تغییر کامل سطح اجرایی سیستم	<i>isolate</i>
systemctl start nginx.service	شروع واحد مشخص شده، مانند یک سرویس یا Target	<i>start name</i>
systemctl stop nginx.service	متوقف کردن واحد مشخص شده	<i>stop name</i>
systemctl reload nginx.service	بارگذاری مجدد فایل پیکربندی واحد مشخص شده بدون توقف کامل آن	<i>reload name</i>
systemctl restart nginx.service	متوقف و دوباره راه‌اندازی کردن واحد مشخص شده؛ برای اعمال تغییرات اساسی یا رفع مشکلات	<i>restart name</i>
systemctl status nginx.service	نمایش وضعیت واحد مشخص شده؛ شامل PID، وضعیت فعال یا غیرفعال بودن و مسیر فایل پیکربندی	<i>status name</i>
systemctl enable nginx.service	پیکربندی واحد برای شروع خودکار هنگام بوت بعدی	<i>enable name</i>
systemctl disable nginx.service	پیکربندی واحد برای عدم شروع خودکار هنگام بوت بعدی	<i>disable name</i>

فصل سوم

فصل ۳

نگهداری و پایداری سیستم لینوکس

پایدار نگه داشتن یک سیستم لینوکس کار ساده‌ای نیست و مجموعه‌ای از وظایف مهم و روزمره را شامل می‌شود. مدیر سیستم باید همیشه مراقب باشد که داده‌ها در امنیت کامل باشند و در صورت بروز **مشکل** بتوان آن‌ها را بازیابی کرد. این یعنی پشتیبان‌گیری منظم (**Backup**) به یکی از وظایف کلیدی تبدیل می‌شود. از طرف دیگر، سیستم باید توانایی پاسخ‌گویی به نیازهای کاربران را داشته باشد؛ بنابراین ظرفیت‌سنجی (**Capacity Planning**) اهمیت ویژه‌ای پیدا می‌کند. علاوه بر این، مدیر سیستم باید همیشه نرم‌افزارهای جدید را نصب کند، برنامه‌های قدیمی را به‌روزرسانی کند، و در عین حال کاربران سیستم را از وضعیت فعلی مطلع سازد. ترکیب این کارهاست که یک سیستم لینوکس را در حالت پایدار و قابل‌اعتماد نگه می‌دارد.

اطلاع‌رسانی به کاربران

ارتباط درست با کاربران سیستم یکی از ارکان اصلی مدیریت موفق به شمار می‌رود. کاربران می‌توانند مدیر سیستم‌های دیگر، برنامه‌نویسان، مشتریان یا حتی افراد عادی باشند. اطلاع‌رسانی صحیح باعث می‌شود اعتماد آن‌ها جلب شود و در مواقع ضروری همکاری بیشتری داشته باشند. بسیاری از سازمان‌ها سیاست‌های مشخصی برای اطلاع‌رسانی دارند؛ مثلاً ارسال ایمیل‌های گروهی، پیامک‌های خودکار یا اطلاعیه روی صفحات داخلی شرکت. اما لینوکس ابزارهای قدرتمند و درونی خود را نیز در اختیار مدیران قرار می‌دهد. این ابزارها نه تنها سریع و کارآمد هستند بلکه به‌صورت مستقیم با سیستم‌عامل و کاربران فعال تعامل دارند.

ابزارهای اطلاع‌رسانی در لینوکس

چندین فایل و دستور مهم در لینوکس وجود دارد که برای ارتباط با کاربران استفاده می‌شوند:

- **etc/issue** : / پیامی که قبل از ورود (login) در ترمینال محلی نمایش داده می‌شود.
- **etc/issue.net** : / پیامی که قبل از ورود از طریق SSH یا اتصال شبکه‌ای نمایش داده می‌شود.
- **etc/motd** : / پیام روز (Message of the Day) که بعد از ورود کاربر نمایش داده می‌شود.
- **sbin/shutdown** : / دستور خاموش یا ریست کردن سیستم، همراه با امکان ارسال پیام به کاربران.
- **usr/bin/notify-send** : / دستور ارسال اعلان گرافیکی (Popup Notification) در محیط‌های دسکتاپ.
- **usr/bin/wall** : / دستور ارسال پیام به همه‌ی کاربران فعال در ترمینال‌ها.

این ابزارها هرکدام در سناریوهای خاصی به کار می‌آیند. مثلاً اگر بخواهید پیامی برای همه کاربران قبل از ورود نمایش داده شود، باید از **etc/issue** یا **etc/issue.net** استفاده کنید. اگر هدفتان پیام‌های زنده و لحظه‌ای باشد، **wall** یا **shutdown** مناسب‌تر هستند.

پیام‌رسانی لحظه‌ای (Fluid Messaging)

یکی از روش‌های مهم ارتباط با کاربران، پیام‌رسانی لحظه‌ای است. این روش بیشتر برای زمانی استفاده می‌شود که یک اتفاق در حال رخ دادن است و کاربران باید بلافاصله در جریان قرار بگیرند. برای نمونه، وقتی سیستم قرار است خاموش شود یا وقتی یک سرویس حیاتی دچار **مشکل** می‌شود، مدیر سیستم باید پیام را به‌طور زنده به کاربران منتقل کند.

پیام‌رسانی لحظه‌ای معمولاً در دو حالت استفاده می‌شود:

۱. **شرایط اضطراری**، مثل خاموشی ناگهانی یا قطع سرویس حیاتی.
۲. **اطلاع‌رسانی تکمیلی**، وقتی که از قبل اطلاعیه داده شده اما نیاز است کاربران فعال دوباره **هشدار** بگیرند.

دستور wall

یکی از کاربردی‌ترین ابزارها برای اطلاع‌رسانی زنده، دستور **wall** است. این دستور پیام شما را برای همه کاربران فعال ارسال می‌کند.

است باعث از دست رفتن داده‌ها یا نارضایتی شدید شود. اما اگر با استفاده از ابزارهایی مثل wall یا shutdown اطلاع‌رسانی شود، کاربران زمان کافی برای ذخیره‌سازی کارهای خود خواهند داشت.

دستور write در لینوکس

در سیستم‌های لینوکس، دستور **write** یک ابزار ساده و در عین حال قدرتمند برای ارسال پیام مستقیم به یک کاربر خاص است. این دستور برای مواقعی استفاده می‌شود که شما به عنوان مدیر سیستم یا حتی یک کاربر بخواهید به فردی مشخص روی همان سیستم پیام فوری ارسال کنید. ساختار کلی دستور به صورت زیر است:

```
write username terminal_id
```

در این دستور، **username** نام کاربری کسی است که می‌خواهید پیام را برایش ارسال کنید و **terminal_id** شناسه ترمینالی است که آن کاربر در حال استفاده از آن است. شناسه ترمینال می‌تواند tty1, tty2 برای ترمینال‌های فیزیکی و pts/0, pts/1 برای ترمینال‌های شبیه‌سازی شده در محیط گرافیکی (GUI) باشد.

مثال استفاده از دستور write

۱. مشاهده کاربران فعال و ترمینال‌های آن‌ها:

ابتدا باید ببینیم چه کاربرانی وارد سیستم شده‌اند و روی کدام ترمینال هستند.

```
$ who
alice pts/1 10:30
bob pts/2 10:35
```

در اینجا کاربر alice روی pts/1 و کاربر bob روی pts/2 فعال است.

۲. ارسال پیام به یک کاربر مشخص:

می‌خواهیم به کاربر alice پیام بفرستیم. ترمینال او pts/1 است.

```
$ write alice pts/1
```

۳. سلام! امروز جلسه فنی ساعت ۳ برگزار می‌شود.

```
Ctrl+D
```

- بعد از تایپ پیام، با زدن **Ctrl+D** پیام ارسال می‌شود.
- کاربر alice پیام شما را فوراً روی ترمینال خود مشاهده می‌کند.

۴. نکات مهم:

- دستور **write** فقط پیام فوری ارسال می‌کند و پیام‌ها ذخیره نمی‌شوند.
- قبل از ارسال، مطمئن شوید پیام‌رسانی ترمینال مقصد فعال است (mesg y).
- اگر پیام‌رسانی غیرفعال باشد (mesg n)، پیام شما ارسال نمی‌شود و پیغام خطا دریافت می‌کنید.

بررسی دسترسی پیام با دستور who -T

قبل از استفاده از **write**، بهتر است بررسی کنید که کاربر مورد نظر امکان دریافت پیام را دارد یا خیر. برای این کار از دستور زیر استفاده می‌شود:

```
who -T
```

این دستور وضعیت دسترسی نوشتن (Write Access) هر کاربر وارد شده به سیستم را نمایش می‌دهد. علامت‌ها به این صورت هستند:

۲. Incremental Backup پشتیبان افزایشی

فقط داده‌هایی که از آخرین عملیات پشتیبان‌گیری تغییر کرده‌اند، کپی می‌شوند. سرعت ایجاد آن بالا و نیاز به فضای کمتر است، اما بازیابی داده‌ها طولانی‌تر است زیرا ابتدا باید **پشتیبان کامل** و سپس تمام پشتیبان‌های افزایشی بازگردانی شوند.

۳. Differential Backup پشتیبان تفاضلی

کپی تمام داده‌هایی که از آخرین پشتیبان کامل تغییر کرده‌اند. سرعت ایجاد آن بین پشتیبان کامل و افزایشی است و برای بازیابی، تنها نیاز به پشتیبان کامل و آخرین پشتیبان تفاضلی دارید.

۴. Snapshot Backup پشتیبان لحظه‌ای

یک روش ترکیبی است که ابتدا یک **کپی کامل** از داده‌ها ایجاد می‌کند و سپس با استفاده از **اشاره‌گرها و لینک‌ها**، تنها تغییرات جدید ذخیره می‌شوند. این روش فضای کمتری مصرف می‌کند و امکان بازگشت به هر نقطه زمانی برای بازیابی کامل سیستم را فراهم می‌کند. برخی نرم‌افزارها مانند **rsync** از این روش استفاده می‌کنند.

پشتیبان‌گیری لحظه‌ای (Snapshot) می‌تواند به صورت **Copy-on-Write** باشد که تنها تغییرات ذخیره می‌شوند، یا به صورت **Split-Mirror** که داده‌ها روی یک دستگاه آینه‌ای نگهداری می‌شوند و در هر پشتیبان‌گیری، کپی کاملی از داده‌ها ایجاد می‌شود.

بررسی بازیابی داده‌ها

وقتی صحبت از پشتیبان‌گیری می‌شود، مرحله‌ای حیاتی وجود دارد که اغلب کمتر مورد توجه قرار می‌گیرد: **بازیابی داده‌ها**. هیچ کس دوست ندارد به این موضوع فکر کند، اما واقعیت این است که در طول عمر یک سیستم، وقوع **حوادث ناخواسته** و نیاز به بازیابی داده‌ها اجتناب‌ناپذیر است. فایل‌ها به اشتباه حذف می‌شوند، هارد دیسک‌ها ممکن است خراب شوند، و فجایع بزرگ گاهی رخ می‌دهند. همانطور که مرگ و مالیات اجتناب‌ناپذیر هستند، بازیابی داده‌ها هم جزو واقعیات دنیای فناوری است. **برنامه‌ریزی بازیابی داده‌ها** باید جزو بخش اصلی استراتژی پشتیبان‌گیری باشد. جزئیات این برنامه به شدت به انتخاب‌های دیگر شما در برنامه پشتیبان‌گیری بستگی دارد، اما در هر حالت، باید برای سناریوهای زیر برنامه‌ریزی شود:

بازیابی فایل‌های منفرد یا چندگانه

این رایج‌ترین سناریو در بازیابی داده‌ها است. کاربران به طور تصادفی فایل‌های خود را حذف می‌کنند یا داده‌های خود را خراب می‌کنند. ارائه یک راهکار که کاربران بتوانند **خودشان فایل‌ها را بازیابی کنند**، بسیار مفید است. همچنین آموزش کاربران در پیشگیری از چنین **مشکلاتی**، بخش مهمی از استراتژی محافظت از داده‌ها است.

بازیابی پارتیشن یا دیسک کامل

اگر ساختارهای مناسب مانند **RAID** داشته باشید، احتمال وقوع چنین **مشکلاتی** کمتر می‌شود، اما همچنان امکان‌پذیر است. در این سناریو، **Recovery Time Objective (RTO)** شما نقش بسیار مهمی دارد و مشخص می‌کند که چه مدت زمان برای بازیابی داده‌ها قابل تحمل است. برنامه بازیابی باید دقیقاً مشخص کند که چطور و چه زمانی پارتیشن‌ها یا دیسک‌های کامل بازیابی خواهند شد.

بازیابی سیستم

این حالت، بدترین سناریوی ممکن است؛ زمانی که کل سیستم و تمام داده‌ها باید بازیابی شوند. بازیابی سیستم به دو روش انجام می‌شود:

۱. **بازیابی کامل سیستم**: شامل بوت کردن سیستم عامل از یک توزیع زنده (Live Distribution) و سپس بازگردانی تمام فایل‌های

لازم برای عملکرد لینوکس، از جمله فایل‌های پیکربندی سیستم و تمام داده‌ها.

۲. **بازیابی پیکربندی و داده‌ها**: ابتدا لینوکس دوباره نصب می‌شود و سپس فایل‌های پیکربندی سیستم و داده‌ها بازیابی می‌شوند.

- باینری‌ها و فایل‌های کتابخانه‌ای و مستندات را به مسیر مناسب منتقل می‌کند
- دسترسی‌ها و مجوزهای فایل‌ها را تنظیم می‌کند

پس از نصب، می‌توانید فایل `tarball` و دایرکتوری نصب موقت را پاک کنید، اما نگه داشتن `tarball` برای احتمالی حذف برنامه (`uninstall`) مفید است.

برای تست برنامه:

```
curl --version
```

مثال عملی: نصب Geany از سورس

مراحل نصب Geany IDE :

۱. با یک کاربر عادی وارد سیستم شوید و مرورگر وب را باز کنید.
۲. به آدرس www.geany.org بروید و `tarball` برنامه را دانلود کنید (نام آن شبیه `geany-##.tar.gz` است).
۳. پس از دانلود، وارد حساب کاربری با دسترسی `super user` شوید.
۴. ابزارهای کامپایل و توسعه را نصب کنید.

RHEL/RedHat/Fedora:
yum groupinstall "Development Tools"
Debian/Ubuntu:
sudo apt-get install build-essential

۵. فایل `tarball` را به دایرکتوری فعلی کپی کنید.

۶. فایل را استخراج کنید:

```
tar -zxvf geany-##.tar.gz
```

اگر فرمت متفاوت است، گزینه‌های `tar` را مطابق فرمت تغییر دهید.

۷. وارد دایرکتوری نصب شوید:

```
cd geany-##
```

۸. فایل‌های `README` و `INSTALL` را بخوانید و بررسی کنید چه وابستگی‌هایی نیاز است.

۹. نصب کتابخانه `GTK2` (و `intltool` برای `Debian/Ubuntu`):

```
sudo apt-get install gtk+2.0 intltool
```

برای RHEL:

```
yum install gtk2-devel
```

۱۰. آماده‌سازی برای کامپایل:

```
./configure
```

در صورت خطای وابستگی، آن‌ها را نصب و دوباره `configure` را اجرا کنید.

۱۱. کامپایل برنامه:

```
make
```

۱۲. نصب نهایی با دسترسی `super user`:

```
sudo make install
```

۱۳. بررسی نسخه برنامه:

collectd یک **Daemon** است که مصرف منابع IT را پایش می‌کند. این نرم‌افزار به زبان C نوشته شده و قابلیت جمع‌آوری داده‌های **Local** و **Remote** را دارد.

- تنظیمات از طریق فایل **collectd.conf** در مسیر **/etc/collectd/** یا **/etc/collectd/** انجام می‌شود.
- گزینه **LoadPlugin** مشخص می‌کند چه پلاگین‌هایی فعال باشند.
- نیاز به ابزارهای نمایش داده‌ها مانند **Front End** دارد.
- اطلاعات بیشتر: [collectd](#)

3. MRTG (Multi Router Traffic Grapher)

MRTG نرم‌افزاری برای جمع‌آوری و گراف شبکه است.

- به زبان **Perl** نوشته شده
- قابلیت گراف گرفتن از تقریباً هر دستگاه شبکه‌ای را دارد
- صفحات **HTML** تولید می‌کند که نمودارهای شبکه را نمایش می‌دهند
- می‌تواند همراه با **RRDTool** استفاده شود
- اطلاعات بیشتر: [MRTG](#)

4. Nagios

Nagios یکی از محبوب‌ترین نرم‌افزارهای پایش منابع است که در دو نسخه عرضه می‌شود:

- **Nagios Core**: رایگان و متن‌باز
- **Nagios XI**: نسخه تجاری

ویژگی‌ها:

- پایش سیستم‌ها، دستگاه‌های شبکه و سرویس‌ها
- قابلیت استفاده از **Plugins** برای تعریف بررسی‌های سفارشی
- نمایش وضعیت سیستم‌ها در یک مرکز کنترل مرکزی (**Centralized Dashboard**)
- ارسال هشدار از طریق ایمیل یا پیامک
- داده‌های جمع‌آوری شده می‌توانند با ابزارهای گرافینگ مانند **PNP4Nagios** و **nagiosgraph** تجسم شوند
- اطلاعات بیشتر: [Nagios](#)
- نسخه کاملاً خودکار: [Nagios Community](#)

5. Icinga

Icinga شاخه‌ای از **Nagios** است که پس از توسعه به دو محصول تقسیم شد:

- **Icinga1**: نسخه اصلی فورک شده از **Nagios**
- **Icinga2**: بازنویسی کامل، با رابط کاربری متفاوت و توسعه سریع‌تر

مزیت‌ها:

- سازگار با **Nagios**
- رابط کاربری بهتر و توسعه سریع‌تر
- اطلاعات بیشتر: [Icinga](#)

6. RRDTool

RRDTool مخفف **Round-Robin Database Tool** است.

```
$ sudo iptraf -i eth0
$ sudo iptraf -s          # نمایش آمار خلاصه
$ sudo iptraf -b          # اجرای بدون رنگ برای لاگ
$ sudo iptraf -L logfile.txt # ذخیره گزارش در فایل
```

lsof

نمایش فایل‌های باز توسط پروسه‌ها: توضیح

```
$ lsof
$ lsof -i :22
$ lsof -u username
$ lsof /var/log/syslog # نمایش پروسه‌های باز روی فایل مشخص
$ lsof -c sshd         # نمایش پروسه‌ها بر اساس نام
$ lsof +D /home/user   # نمایش همه فایل‌های باز در دایرکتوری
```

mpstat

بر اساس هسته‌ها CPU نمایش آمار مصرف: توضیح

```
$ mpstat
$ mpstat -P ALL 2
$ mpstat -u 5 3 # CPU هر ۵ ثانیه، ۳ بار، آمار
$ mpstat -I ALL # نمایش وقایع اینترنت‌راپت
$ mpstat -A     # CPU نمایش تمام آمار
```

mtr

برای بررسی مسیر شبکه traceroute و ping ترکیب: توضیح

```
$ mtr google.com
$ mtr -r -c 5 google.com
$ mtr --report-wide google.com
$ mtr -i 2 google.com # فاصله ۲ ثانیه بین پینگ‌ها
$ mtr -z google.com   # مرتب‌سازی مسیرها بر اساس زمان پاسخ
```

netstat

نمایش پورت‌ها، اتصال‌ها و سوکت‌ها: توضیح

```
$ netstat
$ netstat -tulnp
$ netstat -i
$ netstat -s
$ netstat -rn
$ netstat -an | grep ESTABLISHED
```

ntop

مانیتورینگ ترافیک شبکه از طریق وب: توضیح

```
$ sudo ntop
```

مدیریت حافظه سیستم (System Memory Management)

یکی از مهم‌ترین وظایف هسته لینوکس، مدیریت حافظه (Memory Management) است. این وظیفه به هسته اجازه می‌دهد که نحوه اجرای برنامه‌ها و ابزارهای سیستم در محدوده حافظه موجود را کنترل کند.

هسته علاوه بر مدیریت حافظه فیزیکی (Physical Memory)، قادر است حافظه مجازی (Virtual Memory) نیز ایجاد و مدیریت کند. حافظه مجازی، حافظه‌ای است که به طور واقعی وجود ندارد اما روی هارد دیسک ایجاد شده و مانند حافظه واقعی با آن رفتار می‌شود.

مفهوم Swap Space

برای این منظور، هسته از بخشی از هارد دیسک به نام Swap Space استفاده می‌کند. وقتی یک برنامه نیاز به حافظه بیشتری دارد و RAM پر است، هسته صفحات حافظه (Memory Pages) غیر فعال را به Swap منتقل می‌کند. این فرآیند به نام **Swapping Out** شناخته می‌شود.

وقتی برنامه‌ای به صفحه حافظه‌ای نیاز دارد که در Swap قرار دارد، هسته باید فضای کافی در RAM ایجاد کند، معمولاً با جابجایی (Swapping In) صفحه‌ای دیگر به Swap. واضح است که این عملیات زمان‌بر است و می‌تواند سرعت برنامه را کاهش دهد.

صفحات حافظه (Memory Pages)

حافظه به بلوک‌هایی به نام Memory Pages تقسیم می‌شود. هسته یک جدول حافظه (Page Table) نگهداری می‌کند که نشان می‌دهد کدام صفحات در RAM قرار دارند و کدام صفحات به Swap منتقل شده‌اند.

- **Swapping Out** انتقال صفحات غیر فعال به Swap

- **Swapping In** بازگرداندن صفحات مورد نیاز به RAM

این فرآیند به طور مداوم در طول اجرای سیستم ادامه دارد و یکی از دلایل کاهش عملکرد سیستم در مواقع کمبود RAM است.

مثال عملی

فرض کنید یک سرور لینوکس با ۴ گیگابایت RAM و ۲ گیگابایت Swap دارید. اگر یک برنامه سنگین مثل MySQL یا Apache با بیش از ۶ گیگابایت مصرف حافظه اجرا شود، هسته مجبور است صفحات غیر فعال برنامه‌های دیگر را به Swap منتقل کند. در نتیجه:

- دسترسی به حافظه کندتر می‌شود

- Disk I/O افزایش می‌یابد

- عملکرد کل سیستم کاهش می‌یابد

ابزارهای مرتبط برای پایش حافظه:

- **free -h** نمایش میزان حافظه آزاد و استفاده شده

- **vmstat 2 5** نمایش وضعیت حافظه، پردازنده و Swap هر ۲ ثانیه، ۵ بار

- **sar -r** نمایش گزارش مصرف حافظه به صورت تاریخی

نکات کلیدی مدیریت حافظه

۱. **تنظیم اندازه Swap:** اندازه Swap باید متناسب با RAM و نیاز برنامه‌ها باشد. معمولاً برای سرورها، Swap دو برابر RAM توصیه می‌شود.

۲. **حافظه فشرده (Compressed Memory):** برخی توزیع‌ها از قابلیت **zswap** برای فشرده‌سازی صفحات Swap استفاده می‌کنند، که می‌تواند سرعت سیستم را در هنگام **Swapping** افزایش دهد.

مثال عملی: اگر بخواهید دو نسخه هسته مختلف روی یک سیستم داشته باشید، می‌توانید پوشه‌های جداگانه‌ای مانند `usr/src/linux-4.3.3` و `usr/src/linux-4.4.0` ایجاد کرده و هر بار لینک سمبلیک را به نسخه مورد نظر تغییر دهید. این کار باعث می‌شود بتوانید همزمان چند نسخه هسته را روی سیستم تست کنید بدون اینکه نسخه‌های قبلی حذف شوند.

ایجاد فایل پیکربندی هسته لینوکس (Creating the Kernel Configuration File)

قبل از اینکه بتوانید هسته جدید را از کد منبع کامپایل کنید، لازم است مشخص کنید کدام ویژگی‌های هسته (Kernel Features) را می‌خواهید در فایل باینری هسته جدید (Kernel Binary File) خود داشته باشید. این کار با استفاده از فایل پیکربندی هسته (Kernel Configuration File) انجام می‌شود. فایل پیکربندی هسته، یک فایل متنی (Text File) است که شامل خطوط جداگانه برای هر ویژگی هسته است و در آن، نام ویژگی (Feature Name) و تنظیمات آن (Setting) مشخص شده است. این فایل به صورت پیش‌فرض در مسیر `usr/src/linux/.config` ذخیره می‌شود. اگر شما کد منبع هسته را از قبل نصب کرده‌اید، می‌توانید به این فایل نگاه کنید تا ببینید چه ویژگی‌هایی برای پیکربندی در دسترس هستند.

مثالی از بخشی از فایل `.config` در سیستم Ubuntu :

```
CONFIG_64BIT=y
CONFIG_X86_64=y
CONFIG_X86=y
CONFIG_INSTRUCTION_DECODER=y
CONFIG_PERF_EVENTS_INTEL_UNCORE=y
CONFIG_OUTPUT_FORMAT="elf64-x86-64"
CONFIG_ARCH_DEFCONFIG="arch/x86/configs/x86_64_defconfig"
CONFIG_LOCKDEP_SUPPORT=y
CONFIG_STACKTRACE_SUPPORT=y
CONFIG_HAVE_LATENCYTOP_SUPPORT=y
CONFIG_MMU=y
CONFIG_NEED_DMA_MAP_STATE=y
CONFIG_NEED_SG_DMA_LENGTH=y
CONFIG_GENERIC_ISA_DMA=y
CONFIG_GENERIC_BUG=y
CONFIG_GENERIC_BUG_RELATIVE_POINTERS=y
CONFIG_GENERIC_HWEIGHT=y
CONFIG_ARCH_MAY_HAVE_PC_FDC=y
CONFIG_RWSEM_XCHGADD_ALGORITHM=y
```

همانطور که مشاهده می‌کنید، هر ویژگی با `CONFIG_` شروع می‌شود و مقدار آن می‌تواند `y` (فعال) یا `n` (غیرفعال) باشد.

ایجاد خودکار فایل پیکربندی

تنظیم دستی هر خط از فایل پیکربندی بسیار زمان‌بر و خطاپذیر است. خوشبختانه، می‌توان از اسکریپت‌های خودکار (Automated Scripts) برای ایجاد فایل `.config` استفاده کرد. این اسکریپت‌ها سؤالاتی درباره ویژگی‌های مختلف هسته از شما می‌پرسند و بر اساس پاسخ‌ها، فایل پیکربندی را ایجاد می‌کنند.

دستور `make` در کد منبع، این اسکریپت‌ها را اجرا می‌کند. دستور `make` با استفاده از `targets` مشخص می‌کند کدام اسکریپت اجرا شود. چند `target` کاربرد برای ایجاد فایل پیکربندی وجود دارد:

۷. به روزرسانی هسته: به مرور زمان، هسته‌ها برای رفع اشکال و افزایش امنیت باید به روزرسانی شوند. این کار می‌تواند با دانلود بسته‌های جدید هسته یا اعمال Patch ها انجام شود.

کار با فایل‌های ماژول (Working with Module Files)

در لینوکس، ماژول‌ها بخش بسیار مهمی از هسته سیستم را تشکیل می‌دهند. برای اینکه بتوانید به درستی با ماژول‌ها کار کنید، باید با چند فایل کلیدی آشنا شوید. همانطور که در مراحل قبل دیدید، ماژول‌های مورد نیاز برای پشتیبانی از هسته در مسیر `/lib/modules/` ذخیره می‌شوند. هر نسخه هسته پوشه مخصوص به خود را دارد، به عنوان مثال:

```
/lib/modules/4.3.3
```

این ساختار به شما امکان می‌دهد که برای هر نسخه هسته ماژول‌های جداگانه ایجاد کنید و در صورت نیاز از ماژول‌های سفارشی برای نسخه‌های مختلف هسته استفاده کنید. این موضوع به ویژه زمانی مهم است که سیستم شما چند هسته مختلف را پشتیبانی می‌کند یا می‌خواهید از ماژول‌های آزمایشی استفاده کنید بدون اینکه روی هسته اصلی تاثیر بگذارید.

فایل‌های بارگذاری ماژول‌ها هنگام بوت

ماژول‌هایی که هسته هنگام بوت سیستم بارگذاری می‌کند، در فایل‌های مختلف ذخیره می‌شوند:

- در توزیع‌های **Debian**، این ماژول‌ها در فایل `/etc/modules` لیست شده‌اند.
- در توزیع‌های **Red Hat-based** مانند **Red Hat** و **Fedora**، این ماژول‌ها در پوشه `/etc/modules-load.d/` ذخیره می‌شوند.

در اکثر سیستم‌ها، بسیاری از ماژول‌های سخت‌افزاری به صورت **دینامیک** بارگذاری می‌شوند، زیرا سیستم به طور خودکار دستگاه‌های سخت‌افزاری متصل شده را شناسایی می‌کند. بنابراین، در بسیاری از مواقع، فایل‌های پیکربندی ماژول‌ها تعداد کمی ماژول دارند.

پیکربندی ماژول‌ها

گاهی نیاز است که یک ماژول هسته را با تنظیمات خاص سفارشی کنید، مانند تنظیمات سخت‌افزاری که برای عملکرد درست دستگاه لازم است. پیکربندی‌های ماژول‌ها در فایل `/etc/modules.conf` ذخیره می‌شوند. این فایل شامل پارامترهای مختلف ماژول‌ها و دستورالعمل‌های بارگذاری آن‌ها است. برخی از ماژول‌ها به ماژول‌های دیگر وابسته هستند و باید قبل از آن‌ها بارگذاری شوند. این وابستگی‌ها در فایل `modules.dep` تعریف شده‌اند و در مسیر `/lib/modules/version/` ذخیره می‌شوند، جایی که **version** نسخه هسته است. فرمت هر خط در این فایل به شکل زیر است:

```
modulefilename: dependencyfilename1 dependencyfilename2 ...
```

زمانی که دستور `make modules_install` اجرا می‌شود، برنامه **depmod** به صورت خودکار وابستگی ماژول‌ها را شناسایی کرده و فایل `modules.dep` را ایجاد می‌کند. اگر بعد از آن ماژول جدیدی اضافه یا تغییر دهید، باید به صورت دستی دستور زیر را اجرا کنید تا فایل `modules.dep` به روزرسانی شود:

```
# depmod
```

مدیریت ماژول‌های سفارشی با DKMS

اگر ماژول‌های خود را خارج از ماژول‌های استاندارد هسته کامپایل کنید، با یک مشکل مواجه خواهید شد: هر بار که هسته سیستم به روزرسانی می‌شود، ماژول‌های سفارشی نیز باید دوباره کامپایل شوند تا با هسته جدید سازگار باشند. راه حل این مشکل استفاده از **Dynamic Kernel**

- استفاده می‌شود (/) معمولاً برای ریشه → 1
- برای بقیه فایل سیستم‌ها → 2

مثال عملی

فرض کنید می‌خواهیم یک پارتیشن با UUID مشخص را روی /mnt/data به صورت دائمی مانت کنیم:

```
UUID=1234-ABCD /mnt/data ext4 defaults 0 2
```

این رکورد می‌گوید:

- فایل سیستم با UUID 1234-ABCD
- روی دایرکتوری /mnt/data مانت شود
- نوع فایل سیستم ext4 باشد
- از تنظیمات پیش فرض استفاده شود
- در پشتیبان‌گیری dump لحاظ نشود (0)
- هنگام بوت، توسط fsck بعد از ریشه بررسی شود (2)

بیاید یک مثال کامل‌تر برای فایل /etc/fstab با یک پارتیشن اصلی (/dev/sda1) و یک پارتیشن Swap بنویسیم.

مثال /etc/fstab

```
# <file system>  <mount point>  <type>  <options>  <dump>  <pass>
# پارتیشن ریشه
/dev/sda1  /  ext4  defaults  1  1
# پارتیشن سواپ
/dev/sda2  none  swap  sw  0  0
```

توضیح رکوردها

۱. پارتیشن ریشه (/dev/sda1)

- mount point: / → ریشه سیستم
- type: ext4 → نوع فایل سیستم
- options: defaults → استفاده از تنظیمات پیش فرض
- dump: 1 → پشتیبان‌گیری شود dump می‌توان توسط دستور
- pass: 1 → در هنگام بوت اولویت چک فایل سیستم بالاست (معمولاً ریشه اولین است)

۲. پارتیشن Swap (/dev/sda2)

- mount point: none → سواپ نیازی به دایرکتوری مانت ندارد
- type: swap → نوع فایل سیستم مخصوص سواپ
- options: sw → فعال کردن فضای سواپ
- dump: 0 → پشتیبان‌گیری نشود
- pass: 0 → بررسی نشود هنگام بوت توسط fsck

در این مثال ابتدا UUID فعلی نمایش داده می‌شود، سپس با دستور `uuidgen` UUID جدید ساخته می‌شود. در مرحله بعدی با استفاده از `tune2fs` و گزینه `-U`، UUID جدید روی پارتیشن اعمال می‌شود. در نهایت با اجرای دوباره `blkid` می‌توان تغییر انجام‌شده را بررسی کرد.

ابزارهای مدیریت XFS

سیستم‌فایل XFS یکی از قدرتمندترین فایل‌سیستم‌هاست که به دلیل کارایی بالا و قابلیت مقیاس‌پذیری، در توزیع‌هایی مانند RHEL و RedHat به‌طور گسترده استفاده می‌شود. این فایل‌سیستم به‌ویژه برای کار با فایل‌های بزرگ و محیط‌های سازمانی مناسب است.

ابزار `xfs_admin` برای تغییر ویژگی‌های فایل‌سیستم مانند UUID و برچسب استفاده می‌شود. این ابزار مشابه `e2label` یا `tune2fs` در `ext` عمل می‌کند.

ابزار `xfs_fsr` وظیفه بهینه‌سازی چیدمان فایل‌ها در سیستم‌فایل را دارد. با گذر زمان و حذف و اضافه شدن فایل‌ها، داده‌ها ممکن است به صورت پراکنده در دیسک ذخیره شوند. (fragmentation) این ابزار با بازآرایی داده‌ها به افزایش کارایی کمک می‌کند.

ابزار `xfs_growfs` مکان گسترش اندازه‌ی سیستم‌فایل را فراهم می‌کند. یکی از مزایای مهم XFS این است که می‌توان بدون نیاز به `umount` کردن سیستم‌فایل، آن را گسترش داد. این قابلیت در محیط‌های سازمانی که زمان `downtime` بسیار حیاتی است، اهمیت زیادی دارد. برای مثال اگر یک سرور فایل دارید که همیشه باید در دسترس باشد، می‌توانید به سادگی فضای آن را افزایش دهید بدون اینکه سرویس از دسترس خارج شود.

ابزار `xfs_admin`

ابزار `xfs_admin` برای تغییر ویژگی‌های سیستم‌فایل XFS مانند UUID و برچسب استفاده می‌شود.

مثال‌ها:

- نمایش برچسب فعلی سیستم‌فایل:

```
sudo xfs_admin -l /dev/sda1
```

- تغییر برچسب سیستم‌فایل به "DataPartition":

```
sudo xfs_admin -L DataPartition /dev/sda1
```

- تغییر UUID سیستم‌فایل به "۹-۱۲۳۴۵۶۷۸-abc-def0-1234-56789abcdef0":

```
sudo xfs_admin -U 12345678-9abc-def0-1234-56789abcdef0 /dev/sda1
```

توجه: برای استفاده از `xfs_admin` سیستم‌فایل باید `unmounted` باشد.

ابزار `xfs_fsr`

ابزار `xfs_fsr` برای بهینه‌سازی چیدمان فایل‌ها در سیستم‌فایل XFS و کاهش پراکندگی (fragmentation) استفاده می‌شود.

مثال‌ها:

- بهینه‌سازی تمام فایل‌های سیستم‌فایل‌های XFS متصل:

```
sudo xfs_fsr
```

- بهینه‌سازی یک فایل خاص:

```
sudo xfs_fsr /path/to/file
```

توجه: اجرای `xfs_fsr` ممکن است زمان‌بر باشد و بهتر است در زمان‌های کم‌ترافیک سیستم انجام شود.

بررسی یک آرایه RAID

پس از آنکه یک آرایه RAID ساخته شد، اولین گام مهم این است که مطمئن شویم همه چیز درست کار می‌کند. ساده‌ترین ابزار برای این بررسی، فایل `/proc/mdstat` است. این فایل در لینوکس وضعیت فعلی تمامی آرایه‌های RAID فعال را نشان می‌دهد. برای مثال، اگر یک آرایه RAID سطح ۶ به نام `/dev/md0` ایجاد کرده باشیم، با اجرای دستور زیر می‌توانیم وضعیت آن را ببینیم:

```
# cat /proc/mdstat
```

خروجی نمونه:

```
Personalities: [raid6] [raid5] [raid4]
md0: active raid6 sdi1[3] sdh1[2] sdg1[1] sdf1[0]
      2093056 blocks super 1.2 level 6, 512k chunk, [...]
```

```
unused devices: <none>
```

در این مثال می‌بینیم که:

- آرایه `/dev/md0` به صورت **فعال (active)** در حال کار است.
- چهار پارتیشن `/dev/sdi1`، `/dev/sdh1`، `/dev/sdg1`، `/dev/sdf1` به طور کامل در آرایه حضور دارند.
- سطح RAID برابر 6 است و اندازه **chunk** برابر با ۵۱۲ کیلوبایت تنظیم شده است. این نشانه‌ها می‌گویند که آرایه سالم و آماده‌ی استفاده است.

مفهوم Chunk و Stripe در RAID

یکی از مهم‌ترین مفاهیم در RAID، درک ساختار **Stripe** و **Chunk** است.

- **Stripe نواری**: (زمانی که داده روی دیسک‌ها توزیع می‌شود، هر قطعه از داده در قالب یک نوار به چندین دیسک تقسیم می‌شود).
 - **Chunk قطعه**: (کوچک‌ترین واحد داده در هر دیسک داخل آن Stripe است).
- در مثال بالا، اندازه `Chunk = 512KB` است. این یعنی هر بار که داده‌ای نوشته می‌شود، به بلوک‌هایی ۵۱۲ کیلوبایتی تقسیم شده و سپس به دیسک‌ها توزیع می‌گردد.
- هرچه اندازه **Chunk** بزرگ‌تر باشد، برای فایل‌های بزرگ عملکرد بهتری خواهیم داشت (مثل فایل‌های ویدئویی یا دیتابیس‌های حجیم). در مقابل، برای فایل‌های کوچک، **Chunk** کوچک‌تر می‌تواند کارایی را بهبود دهد.

بررسی جزئیات بیشتر با mdadm

علاوه بر فایل `/proc/mdstat`، می‌توان از حالت **Miscellaneous** ابزار `mdadm` برای بررسی دقیق‌تر استفاده کرد. مثال زیر، اطلاعات کامل آرایه `/dev/md0` را نمایش می‌دهد:

```
# mdadm --misc --detail /dev/md0
```

یا حتی کوتاه‌تر:

```
# mdadm --detail /dev/md0
```

برای دیدن دیسک‌های **Nvme** متصل و namespace های آنها:

```
sudo nvme list
```

خروجی مشابه خواهد بود با جدولی شامل Device، Namespace(s)، Size، و مسیرها.

دستور id-ctrl — Identify Controller

برای گرفتن اطلاعات کامل کنترلر **Nvme**:

```
sudo nvme id-ctrl /dev/nvme0
```

برای نمایش به شکل خوانا با هدر:

```
sudo nvme id-ctrl -H /dev/nvme0
```

خروجی شامل فیلدهایی مانند mn (Model Name)، sn (Serial Number)، ver (Firmware Version)، nn (Number of Namespaces) و غیره خواهد بود.

دستور id-ns — Identify Namespace

برای گرفتن اطلاعات یک namespace خاص، مثلاً namespace شماره ۱:

```
sudo nvme id-ns /dev/nvme0n1
```

یا با گزینه خوانا:

```
sudo nvme id-ns -H /dev/nvme0n1
```

خروجی فیلدهایی مانند nsze (Namespace Size)، flbas (LBA Format)، nuse، ncap و غیره را نمایش می‌دهد.

ایجاد و حذف Namespace

اگر دستگاه **Nvme** اجازه دهد، می‌توانی namespace جدید بسازی یا حذف کنی:

• ایجاد Namespace:

```
sudo nvme create-ns /dev/nvme0 -s 100G -c 100G
```

در این مثال یک namespace با ظرفیت ۱۰۰ گیگابایت ساخته می‌شود.

• حذف Namespace:

فرض کن namespace با شناسه 3 را بخواهی حذف کنی:

```
sudo nvme delete-ns /dev/nvme0 -n 3
```

اتصال و جدا کردن (attach / detach) Namespace

برای اتصال یک namespace به کنترلر:

```
sudo nvme attach-ns /dev/nvme0 -n 2 -c 0
```

در این مثال، namespace با شناسه ۲ به کنترلر شماره ۰ متصل می‌شود.

برای جدا کردن (detach):

```
sudo nvme detach-ns /dev/nvme0 -n 2
```

درک و پیاده‌سازی این روش ذخیره‌سازی قبل از نیاز واقعی به آن اقدامی بسیار هوشمندانه است. در اینجا با مفاهیم اصلی LVM، ساختار آن و همچنین مراحل عملی پیاده‌سازی آشنا خواهیم شد.

آشنایی با LVM

مدیریت حجم منطقی (Logical Volume Management – LVM) یک فناوری است که در سیستم‌های لینوکس به‌طور گسترده برای مدیریت دیسک و پارتیشن‌ها به کار می‌رود. این فناوری به شما اجازه می‌دهد چندین **پارتیشن** یا حتی چندین **دیسک فیزیکی** را به یکدیگر متصل کرده و به‌صورت یکپارچه از آنها استفاده کنید. نتیجه این کار یک فضای ذخیره‌سازی منعطف است که می‌توان آن را **فرمت** کرد، روی ساختار دایرکتوری مجازی لینوکس **مونت (mount)** نمود، داده‌ها را روی آن ذخیره کرد و حتی در آینده آن را تغییر اندازه داد.

ویژگی مهم LVM این است که یک **لایه انتزاعی (Abstraction Layer)** ایجاد می‌کند تا چندین پارتیشن یا دیسک فیزیکی به شکل یک واحد مستقل دیده شوند. این لایه باعث می‌شود که مدیریت داده‌ها ساده‌تر و بسیار منعطف‌تر از پارتیشن‌بندی سنتی باشد.

به عنوان مثال، فرض کنید در ابتدا یک دیسک ۵۰۰ گیگابایتی دارید و یک **حجم منطقی** روی آن ساخته‌اید. بعد از مدتی نیاز به فضای بیشتر پیدا می‌کنید. در سیستم‌های سنتی مجبور می‌شدید پارتیشن‌های جدید بسازید یا حتی داده‌ها را به دیسک دیگری منتقل کنید. اما در LVM تنها کافی است یک دیسک جدید (مثلاً ۱ ترابایت) اضافه کنید و آن را به همان حجم منطقی متصل نمایید. در نتیجه بدون نیاز به جابه‌جایی داده‌ها، فضای ذخیره‌سازی شما افزایش پیدا می‌کند.

اجزای اصلی LVM

LVM از سه بخش اصلی تشکیل شده است که هر کدام نقشی کلیدی در ایجاد و نگهداری حجم‌های منطقی دارند:

۱. حجم فیزیکی (Physical Volume – PV)

حجم فیزیکی (PV) اولین مرحله در ساخت LVM است. یک PV معمولاً یک **پارتیشن دیسک** یا حتی کل یک **هارد دیسک** است که با دستور `bin/pvcreate` آماده استفاده در LVM می‌شود. هنگام اجرای این دستور، داده‌های ساختاری LVM شامل **برچسب حجم (Volume Label)** و **متادیتا (Metadata)** روی پارتیشن نوشته می‌شوند. این متادیتا اطلاعاتی مانند شناسه یکتا، وضعیت PV و تنظیمات پایه را ذخیره می‌کند.

نکته مهم این است که پس از ایجاد یک PV، نمی‌توانید آن را به‌طور مستقیم برای ذخیره داده استفاده کنید. بلکه ابتدا باید آن را وارد یک **گروه حجم (VG)** کنید.

۲. گروه حجم (Volume Group – VG)

گروه حجم (VG) مجموعه‌ای از یک یا چند PV است که با دستور `bin/vgcreate` ساخته می‌شود. VG مانند یک **استخر ذخیره‌سازی (Storage Pool)** عمل می‌کند که از آن برای ایجاد حجم‌های منطقی استفاده می‌کنیم. ویژگی مهم VG این است که شما می‌توانید چندین PV را به آن اضافه کنید و در نتیجه یک فضای ذخیره‌سازی بزرگ و یکپارچه بسازید.

به عنوان مثال، اگر سه دیسک ۵۰۰ گیگابایتی داشته باشید، می‌توانید آنها را به یک VG اضافه کنید و یک فضای کلی ۱.۵ ترابایتی ایجاد نمایید. سپس حجم‌های منطقی خود را بر اساس این فضا بسازید. هر PV فقط می‌تواند عضو یک VG باشد، اما یک دیسک می‌تواند چندین پارتیشن داشته باشد که هر کدام به VG های مختلف اختصاص داده شوند.

۳. حجم منطقی (Logical Volume – LV)

جدول کامل پرتکل‌ها و پورت‌ها

توضیحات کاربردی	TCP/UDP پورت	پروتکل / سرویس	لایه شبکه
انتقال صفحات وب بدون رمزگذاری	80 TCP	HTTP	(Application) لایه کاربردی
TLS/SSL انتقال صفحات وب امن با	443 TCP	HTTPS	
انتقال فایل	20 TCP (Data), 21 TCP (Control)	FTP	
SSH انتقال امن فایل با	22 TCP	SFTP	
دسترسی امن ریموت به سرور	22 TCP	SSH	
دسترسی ریموت غیر امن (امروزه کمتر استفاده می‌شود)	23 TCP	Telnet	
ارسال ایمیل	25 TCP	SMTP	
SSL با رمزگذاری SMTP	465 TCP	SMTPS	
دریافت ایمیل با مدیریت پوشه‌ها	143 TCP	IMAP	
SSL امن با IMAP	993 TCP	IMAPS	
دریافت ایمیل ساده	110 TCP	POP3	
SSL امن با POP3	995 TCP	POP3S	
و بالعکس IP ترجمه نام دامنه به	53 TCP/UDP	DNS	
به کلاینت‌ها IP تخصیص خودکار	67/68 UDP	DHCP	
هماهنگی زمان در شبکه	123 UDP	NTP	
مدیریت و مانیتورینگ شبکه	161/162 UDP	SNMP	
مدیریت دایرکتوری شبکه	389 TCP/UDP	LDAP	
SSL امن با LDAP	636 TCP	LDAPS	
VoIP پروتکل سیگنالینگ	5060 TCP/UDP	SIP	
SSL امن با SIP	5061 TCP	SIPS	
با بهینه‌سازی HTTP نسخه جدید	443 TCP	HTTP/2	
MySQL اتصال به دیتابیس	3306 TCP	MySQL	
PostgreSQL اتصال به دیتابیس	5432 TCP	PostgreSQL	
و کش NoSQL پایگاه داده	6379 TCP	Redis	
IoT پروتکل پیام‌رسانی سبک	1883 TCP	MQTT	
SSL امن با MQTT	8883 TCP	MQTTs	
پروتکل اتصال گرا، تضمین تحویل داده‌ها	—	TCP	(Transport) لایه انتقال
پروتکل بدون اتصال، سریع برای صدا و ویدئو	—	UDP	
پروتکل انتقال پیام مقاوم برای همزمانی	—	SCTP	
IPv4 و IPv6 مسیریابی داده‌ها،	—	IP	(Network) لایه شبکه
(ping) کنترل پیام و تست وضعیت شبکه	—	ICMP	
LAN در شبکه MAC به IP تبدیل آدرس	—	ARP	
MAC بر اساس IP، پیدا کردن ARP معکوس	—	RARP	
multicast مدیریت گروه‌های	—	IGMP	
شبکه سیم‌دار، استاندارد کابل مسی و فیبر	—	Ethernet	(Physical) لایه فیزیکی
شبکه بی‌سیم، شامل باندهای ۲.۴ و ۵ گیگاهرتز	—	Wi-Fi (802.11)	
انتقال داده با نور، سرعت بالا و فاصله طولانی	—	Fiber-optic	

mtr www.google.com

My traceroute [v0.85]

Host	Loss%	Snt	Last	Avg	Best	Wrst	StDev
1. 10.0.2.2	0.0%	24	7.0	1.5	0.0	7.0	1.7
2. homeportal	0.0%	24	5.4	5.3	0.8	17.4	3.9
3. 162-194-188-2.lightspeed.ip	8.3%	23	14.15	14.08	14.43	14.33	3.6

ویژگی‌های مهم mtr:

- نمایش درصد بسته‌های از دست رفته (Loss%) در هر هاپ
- نمایش زمان آخرین پکت، میانگین، بهترین و بدترین زمان
- بروزرسانی لحظه‌ای اطلاعات شبکه
- کمک به شناسایی مشکلات عملکردی در مسیر شبکه

آزمایش اتصال کلاینت و سرور

گاهی اوقات، ارسال پکت‌های ICMP کافی نیست و نیاز دارید اتصال واقعی داده‌ها را شبیه‌سازی کنید. دستور nc (Netcat) برای این کار عالی است.

Netcat می‌تواند هم به عنوان کلاینت و هم سرور عمل کند و به شما اجازه می‌دهد داده‌ها را از طریق شبکه ارسال و دریافت کنید.

گزینه

توضیح

4-	استفاده از IPv4
6-	استفاده از IPv6
b-	اجازه ارسال پکت‌های Broadcast
C-	ارسال CRLF در انتهای هر خط
D-	فعال کردن حالت Debug
d-	عدم خواندن از ورودی استاندارد
h-	نمایش راهنما
I len-	اندازه بافر دریافت TCP
i int-	فاصله زمانی بین ارسال و دریافت
k-	ادامه گوش دادن بعد از بسته شدن اتصال
l-	گوش دادن به یک پورت مشخص
n-	غیرفعال کردن جستجوی DNS
O len-	اندازه بافر خروجی TCP
p port-	تعیین پورت مبدأ
q sec-	تعداد ثانیه انتظار پس از EOF
s addr-	تعیین آدرس IP اینترنتیس ارسال کننده
t-	اجازه اجرای Telnet اسکریپتی

۴. کدهای پاسخ عملکردی (Action Response Codes)

- کدهای خطا به جزئیات دقیق اشاره نمی‌کنند و تنها یک ایده کلی از مشکل ارائه می‌دهند.
- کدهای اطلاعاتی برای نمایش اطلاعات اضافی درباره یک دستور استفاده می‌شوند.
- کدهای سرویس وضعیت سرویس SMTP در اتصال را نشان می‌دهند.
- کدهای عملکردی نتیجه تلاش سرور برای اجرای دستورات MAIL، RCPT و DATA را نشان می‌دهند و به کلاینت اطلاع می‌دهند که چه اقدام بعدی انجام شود.

جدول نمونه کدهای SMTP:

کد	دسته‌بندی	توضیح
500	خطا	خطای نحوی، دستور شناسایی نشد
501	خطا	خطای نحوی در پارامترها
502	خطا	دستور پیاده‌سازی نشده
503	خطا	ترتیب دستورات اشتباه است
504	خطا	پارامتر دستور پیاده‌سازی نشده
211	اطلاعاتی	وضعیت سیستم یا کمک سیستم
214	اطلاعاتی	پیام راهنما
220	سرویس	سرویس آماده است
221	سرویس	سرویس در حال بسته شدن است
421	سرویس	سرویس در دسترس نیست
250	عملکردی	دستور ایمیل موفقیت‌آمیز بود
251	عملکردی	کاربر محلی نیست، پیام به مسیر هدایت می‌شود
354	عملکردی	شروع ورود پیام، پایان با <CRLF>.<CRLF>
450	عملکردی	عملیات انجام نشد، Mailbox در دسترس نیست
451	عملکردی	عملیات متوقف شد، خطا در پردازش
452	عملکردی	عملیات انجام نشد، فضای ذخیره‌سازی کافی نیست
550	عملکردی	عملیات انجام نشد، Mailbox غیرقابل دسترس
551	عملکردی	کاربر محلی نیست، مسیر جایگزین پیشنهاد شد
552	عملکردی	عملیات متوقف شد، حجم ذخیره‌سازی بیش از حد
553	عملکردی	عملیات انجام نشد، نام Mailbox مجاز نیست
554	عملکردی	تراکشن شکست خورد

نکته عملی: دانستن این کدها برای **عیب‌یابی ایمیل‌ها** حیاتی است. اگر یک پیام ایمیل بازگردد (Bounce)، معمولاً کد پاسخ دقیق نیز همراه با ایمیل ارائه می‌شود تا علت شکست مشخص شود.

Extended SMTP (ESMTP)

با گذر زمان، محدودیت‌های SMTP استاندارد آشکار شد. به جای جایگزینی پروتکل قدیمی که در سطح جهانی استفاده می‌شد، Extended SMTP یا ESMTP معرفی شد تا ویژگی‌های جدید را به SMTP اضافه کند و در عین حال سازگاری با پروتکل قدیمی حفظ شود.

ویژگی‌های ESMTP:

- اضافه کردن دستورات اختیاری برای قابلیت‌های ویژه
- کلاینت و سرور باید **مذاکره کنند** که کدام دستورات استفاده شود

فصل ۷

سیستم نام دامنه (DNS) و نرم افزار BIND

سیستم نام دامنه (DNS) یک پروتکل شبکه‌ای است که در قلب بسیاری از فعالیت‌های اینترنتی قرار دارد. این سیستم وظیفه دارد آدرس‌های IP را به سیستم‌های مختلفی که خدمات ارائه می‌دهند اختصاص دهد و برای انجام این کار از یک پایگاه داده توزیع شده استفاده می‌کند. برای سیستم‌های لینوکس، نرم افزار BIND یکی از محبوب‌ترین بسته‌ها برای راه اندازی خدمات DNS است. راه اندازی DNS از طریق BIND می‌تواند فرآیندی پیچیده باشد، زیرا نیاز به ویرایش فایل‌های پیکربندی، ایجاد و مدیریت زون‌ها، و همچنین تأمین امنیت سرور DNS و تراکنش‌های مختلف آن دارد. در این فصل، تمامی این موضوعات و موارد مرتبط با آن پوشش داده شده است.

پیکربندی یک سرور DNS

تا حدی، شما باید از DNS تشکر کنید که می‌توانید آدرس یک وبسایت، مانند (www.ivytech.edu) یا نسخه کوتاه‌تر آن، مانند (ivytech.edu) را در مرورگر وارد کرده و مرورگر شما را به سایت مورد نظر برساند. در این بخش، شما با نحوه عملکرد DNS، نصب نرم افزار BIND برای ارائه خدمات DNS، فایل‌های پیکربندی مختلف BIND، راه اندازی و توقف آن، تنظیم لاگ‌گیری در BIND، و حتی پیاده‌سازی یک سرور DNS محلی فقط با حافظه کش آشنا خواهید شد.

درک DNS و BIND

قبل از اینکه وارد نحوه نصب و پیکربندی BIND شویم، بهتر است ابتدا نحوه کار DNS را درک کنیم. DNS در سال ۱۹۸۳ طراحی شد و ساختاری پیچیده اما منظم دارد که آزمون زمان را به خوبی پشت سر گذاشته است.

نگاه کلی به DNS

معمولاً وقتی کسی می‌خواهد به یک وبسایت دسترسی پیدا کند، مرورگر را باز کرده و حروف یا کلمات را که با نقطه جدا شده‌اند، در نوار آدرس وارد می‌کند، مانند:

www.ivytech.edu

این رشته‌های جدا شده با نقطه، یک نام دامنه کامل یا FQDN (Fully Qualified Domain Name) را تشکیل می‌دهند. FQDN ها آدرس‌های قابل فهم برای انسان هستند که افراد از طریق آن‌ها به خدمات مختلف مانند صفحات وب دسترسی پیدا می‌کنند. اما سیستم‌های متصل به شبکه، مانند اینترنت، از FQDN ها استفاده نمی‌کنند؛ آن‌ها از آدرس‌های IP برای شناسایی یکدیگر استفاده می‌کنند. بنابراین نیاز است که FQDN ها به IP و برعکس تبدیل شوند. این نیاز معمولاً توسط DNS برآورده می‌شود. فرآیند تبدیل بین FQDN یک سیستم و آدرس IP آن، حل نام یا Name Resolution نامیده می‌شود.

در گذشته، فایل محلی `/etc/hosts` در سیستم‌ها برای حل نام استفاده می‌شد. هنوز هم در شبکه‌های کوچک محلی که شامل تعداد کمی سیستم هستند، می‌توان از این فایل برای حل نام‌ها استفاده کرد.

اگر DNS تنها از یک پایگاه داده در یک مکان استفاده می‌کرد، نتایج فاجعه‌بار می‌شد؛ شبکه‌های بزرگ هنگام انتظار برای حل نام‌ها دچار کندی می‌شدند. خوشبختانه، پایگاه داده DNS دارای ساختار توزیع شده و قابلیت مقیاس پذیری بالا است که عملکرد لازم برای حل نام‌ها را فراهم می‌کند.

ساختار پایگاه داده DNS

ساختار پایگاه داده DNS که به آن فضای نام دامنه یا Domain Name Space نیز گفته می‌شود، به صورت یک درخت وارونه یا معکوس است. این ساختار تا حدودی شبیه به ساختار دایرکتوری مجازی لینوکس است.

در DNS، یک میزبان وب ممکن است با نام میزبان و دامنه‌ها که با نقطه از هم جدا شده‌اند، مانند www.example.com مشخص شود، در حالی که در ساختار دایرکتوری لینوکس، فایل‌ها و دایرکتوری‌ها با اسلش‌های مستقیم / جدا می‌شوند، مانند:

`/home/rich/filea.txt`

نصب و ابزارهای BIND

قبل از پیکربندی، باید نرم‌افزار BIND روی سیستم نصب شده باشد. حتی اگر فقط یک سرور **caching-only** پیکربندی می‌کنید، بهتر است ابزارهای کمکی BIND مانند بسته **bind-utils** نیز نصب شوند. این ابزارها شامل ابزارهایی مانند **dig**، **nslookup** و **named-checkconf** هستند که برای تست و عیب‌یابی ضروری‌اند. در برخی توزیع‌ها، فایل‌های پیکربندی BIND از پیش برای ارائه سرویس **DNS caching-only** به کلاينت‌ها تنظیم شده‌اند و شما تنها نیاز به تغییرات جزئی دارید. در توزیع‌های دیگر، ممکن است چندین فایل پیکربندی نیاز به تغییر داشته باشند تا سرور **caching-only** راه‌اندازی شود.

نکته عملی: برخی توزیع‌ها مانند **Ubuntu** با **dnsmasq** پیش‌نصب شده و فعال هستند. اگر قصد استفاده از BIND دارید، ممکن است تداخل و نتایج غیرمنتظره‌ای در کش رخ دهد. بنابراین قبل از راه‌اندازی BIND، باید **dnsmasq** غیرفعال شود.

نصب BIND

در ابتدا باید بسته‌های مورد نیاز را نصب کنیم. بسته به توزیع لینوکس، دستور نصب متفاوت است:

در توزیع‌های Debian/Ubuntu :

```
sudo apt update
sudo apt install bind9 bind9utils bind9-doc -y
```

در توزیع‌های RHEL/CentOS/Fedora :

```
sudo dnf install bind bind-utils -y
```

پس از نصب، سرویس BIND به صورت پیش‌فرض فعال است، ولی بهتر است وضعیت آن را بررسی کنیم:

```
sudo systemctl status named
```

اگر سرویس فعال نبود:

```
sudo systemctl start named
sudo systemctl enable named
```

ساختار فایل‌های پیکربندی

BIND از چند فایل اصلی برای پیکربندی استفاده می‌کند:

فایل	توضیح
<i>etc/bind/named.conf</i>	فایل اصلی پیکربندی
<i>etc/bind/named.conf.options</i>	تنظیمات عمومی، مانند forwarders
<i>etc/bind/named.conf.local</i>	تعریف zone های محلی
<i>var/cache/bind/</i>	محل کش DNS

پیکربندی DNS caching-only

یک **DNS caching-only** فقط درخواست‌ها را ذخیره و پاسخ می‌دهد و معمولاً برای سرعت‌بخشی به درخواست‌های DNS شبکه داخلی استفاده می‌شود.

فصل ۸

استفاده از لینوکس به عنوان وب سرور

یکی از رایج‌ترین کاربردهای لینوکس در محیط‌های سروری، استفاده از آن به عنوان **وب سرور (web server)** است. وب سرورهای لینوکس بر اینترنت غالب هستند و همچنین در میزبانی **برنامه‌های سازمانی (corporate intranet applications)** نیز بسیار محبوب‌اند. از ارائه صفحات استاتیک وب گرفته تا میزبانی **برنامه‌های وب دینامیک (dynamic web applications)**، اغلب روی پلتفرم لینوکس با سرعت و کارایی بالاتری اجرا می‌شوند.

این فصل به شما نشان می‌دهد چگونه سیستم لینوکس خود را برای پشتیبانی از انواع محیط‌های وب سرور پیکربندی کنید. ابتدا توضیح می‌دهد **وب سرور چیست** و چه نرم‌افزارهایی برای تبدیل لینوکس به وب سرور لازم است. سپس به نصب و پیکربندی نرم‌افزار **Apache** می‌پردازد که محبوب‌ترین وب سرور حال حاضر است. از آنجا که **Apache** بسیار انعطاف‌پذیر است، پیکربندی آن ممکن است کمی پیچیده باشد، اما این فصل مراحل لازم برای راه‌اندازی وب‌سایت را به صورت مرحله به مرحله شرح می‌دهد.

پس از آن، **وب سرور پراکسی Squid** معرفی می‌شود. استفاده از پراکسی وب سرور در محیط‌هایی که نیاز به **کشینگ صفحات وب (web page caching)** برای صرفه‌جویی در پهنای باند شبکه و افزایش کارایی دارند، بسیار مفید است. در نهایت، **وب سرور nginx** بررسی می‌شود؛ محصولی که به سرعت در حال محبوب شدن است و ویژگی‌های جدید و جذابی ارائه می‌کند.

وب سرور چیست؟

قدرت اینترنت به انتقال آزاد و باز اطلاعات برمی‌گردد. در اوایل دوران اینترنت، دسترسی به فایل‌های داده روی سیستم‌های راه دور نیازمند داشتن حساب کاربری (**user login account**) و استفاده از برنامه‌های پیچیده انتقال فایل (**file transfer**) بود. برای تسهیل به اشتراک‌گذاری سریع و باز اطلاعات، نیاز به روشی جدید بود.

HTTP و تولد وب مدرن

- پروتکل **Hypertext Transfer Protocol (HTTP)** توسعه یافت تا انتقال داده‌ها بین سیستم‌ها به صورت ناشناس و ساده انجام شود.
- برای داده‌های عمومی که نیاز به حفاظت ندارند، **HTTP** اجازه می‌دهد **کلاینت** به سرور متصل شود. فایل‌های داده را دریافت کند و سپس ارتباط را قطع کند.
- این روش باعث شد سرعت دریافت داده‌ها از سیستم‌های راه دور افزایش یابد و به محبوب شدن اینترنت به عنوان محیطی برای به اشتراک‌گذاری اطلاعات کمک کرد.

HTML و سازماندهی داده‌ها

با وجود **HTTP**، هنوز مشکل این بود که پیدا کردن اینکه چه اطلاعاتی در کدام فایل متنی ذخیره شده، و چه فایل‌هایی باید از چه سروری دریافت شود، دشوار بود. برای حل این مشکل، نیاز به روشی برای لینک دادن فایل‌های مرتبط و ایجاد استاندارد جدید بود.

Hypertext Markup Language (HTML) دقیقاً این کار را انجام داد:

- **HTML** به نویسندگان اجازه می‌دهد اسناد خود را با استفاده از لینک‌های ابرمتنی (**hyperlinks**) به اسناد دیگر متصل کنند.
 - لینک‌ها فایل‌های راه دور را با مشخص کردن نام سرور و مسیر فایل ارجاع می‌دهند.
 - نویسندگان تنها کافی بود لینک‌ها را در متن خود قرار دهند تا کاربران بتوانند به آسانی از یک سند به سند دیگر بروند.
- همچنین **HTML** امکان فرمت‌دهی متن را فراهم می‌کند:

کدهای رایج پاسخ HTTP

در جدول زیر تعدادی از کدهای استاندارد پاسخ سرور HTTP و معنای آنها آورده شده است:

کد وضعیت	پیام متنی	توضیح
100	Continue	ادامه درخواست
101	Switch Protocols	تغییر پروتکل
102	Processing Request	پردازش درخواست
200	OK	موفقیت آمیز
201	Created	ایجاد شد
202	Accepted	پذیرفته شد
206	Partial Content	محتوای جزئی
207	Multi-Status	وضعیت چندگانه
208	Already Reported	قبلاً گزارش شده
226	IM Used	از IM استفاده شد
300	Multiple Choices	گزینه‌های متعدد
301	Moved Permanently	منتقل شد
302	Found	پیدا شد
303	See Other	مشاهده مورد دیگر
304	Not Modified	تغییر نکرده
305	Use Proxy	از پراکسی استفاده شود
307	Temporary Redirect	تغییر مسیر موقت
308	Permanent Redirect	تغییر مسیر دائمی
400	Bad Request	درخواست اشتباه
401	Unauthorized	غیرمجاز
403	Forbidden	ممنوع
404	Not Found	پیدا نشد
405	Method Not Allowed	متد مجاز نیست
406	Not Acceptable	غیرقابل قبول
408	Request Timeout	زمان درخواست تمام شد
409	Conflict	تعارض
410	Gone	حذف شده
500	Internal Server Error	خطای داخلی سرور
501	Not Implemented	پیاده‌سازی نشده

پس از نصب بسته‌های سرور و وابستگی‌ها، می‌توانید فرآیند پیکربندی را شروع کنید. قبل از پیکربندی اشتراک فایل‌ها، مرور اجمالی بر دایرکتوری‌ها و ابزارهای Samba توصیه می‌شود.

مثال عملی: ایجاد یک Samba Share ساده

فرض کنید می‌خواهید یک پوشه /srv/samba/shared را بین کاربران لینوکس و ویندوز به اشتراک بگذارید.

۱. نصب بسته‌های مورد نیاز:

```
sudo yum install samba samba-client cifs-utils
```

۲. ایجاد پوشه به اشتراک گذاشته شده:

```
sudo mkdir -p /srv/samba/shared
sudo chown nobody:nogroup /srv/samba/shared
sudo chmod 0775 /srv/samba/shared
```

۳. افزودن اشتراک در فایل پیکربندی /etc/samba/smb.conf:

```
[shared]
path = /srv/samba/shared
browsable = yes
writable = yes
guest ok = yes
read only = no
```

۴. راه‌اندازی سرویس Samba :

```
sudo systemctl start smb
sudo systemctl enable smb
sudo systemctl restart nmb
```

۵. دسترسی کاربران ویندوز:

- از Run ویندوز: \\server-ip\shared
- فایل‌ها را مشاهده و ویرایش کنید.

بررسی دایرکتوری‌ها و فایل‌های Samba

علاوه بر دایرکتوری مستندات (Documentation Directory) که پیش‌تر توضیح داده شد، سه دایرکتوری اصلی Samba وجود دارند که فایل‌های مختلف Samba در آن‌ها قرار دارند. یکی از این دایرکتوری‌ها، محل ذخیره فایل‌های لاگ Samba (Log Files) است: `./var/log/samba`

Samba همچنین از دایرکتوری `/var/lib/samba/` و زیرشاخه‌های آن برای ذخیره اطلاعاتی مانند پایگاه داده کاربران (User Database) استفاده می‌کند. به عنوان مثال، اطلاعات حساب کاربران Samba می‌تواند در یکی از پایگاه‌های داده `smbpasswd`، `tdbsam` یا `ldbapsam` ذخیره شود که معمولاً در همین دایرکتوری قرار دارند.

- برای پیاده‌سازی‌های کوچک Samba (کمتر از ۲۵۰ کاربر)، پایگاه داده `tdbsam` توصیه می‌شود.
- پیاده‌سازی‌های بزرگ‌تر باید از پایگاه داده `ldbapsam` همراه با (Active Directory دایرکتوری فعال) یا LDAP (Lightweight Directory Access Protocol) پروتکل سبک دسترسی به دایرکتوری (استفاده کنند).
- پایگاه داده `smbpasswd` در حال منسوخ شدن است و تنها برای سازگاری با نسخه‌های قدیمی Samba استفاده می‌شود.

راه‌اندازی سرویس Samba در زمان بوت

برای اینکه سرویس Samba هنگام بوت شدن سیستم خودکار اجرا شود، از دستور زیر استفاده می‌کنیم:

```
# systemctl enable smb
```

نکته امنیتی: در برخی توزیع‌ها ممکن است نیاز به تنظیم SELinux Booleans برای عملکرد صحیح Samba باشد. برای تست، می‌توان به‌طور موقت SELinux را غیرفعال کرد:

```
# sestatus
# nano /etc/selinux/config
# SELINUX=disabled
# reboot
```

پیکربندی فایروال برای Samba

برای عملکرد صحیح Samba با کلاینت‌ها، فایروال باید پورت‌های مورد نیاز را باز کند. برخی از پورت‌های مهم Samba در جدول زیر آمده‌اند:

توضیح	پروتکل‌ها	شماره پورت
DNS داخلی	TCP, UDP	53
Kerberos	TCP, UDP	88
End Point Resolution	TCP	135
خدمات NetBIOS	TCP, UDP	137-139
SMB over TCP	TCP	445
LDAP	TCP, UDP	389
Kerberos kpasswd	TCP, UDP	464
LDAP over SSL (LDAPS)	TCP	636
Samba Web Administration Tool (SWAT)	TCP, UDP	901
Dynamic RPC Ports	TCP	1024-5000
Microsoft Global Catalog	TCP	3268-3269
Multicast DNS	TCP, UDP	5353

برای مشاهده پورت‌های استفاده شده توسط Daemon های Samba، از دستور ss استفاده می‌کنیم:

```
# systemctl status smb | grep PID
# ss -utlpn | grep <PID>
```

پیکربندی کلاینت Samba

پس از آماده شدن اشتراک‌های سرور، می‌توان کلاینت لینوکس را تنظیم کرد. مراحل کلی به شرح زیر است:

۱. نصب یا به‌روزرسانی بسته Samba client و ابزارهای جانبی.
۲. ایجاد حساب کاربری محلی مشابه سرور (همان UID).
۳. ورود به سرویس Samba با smbclient و مشاهده اشتراک‌ها.
۴. ایجاد Mount Point برای اشتراک.
۵. نصب موقت اشتراک با دستور mount.

```
option routers 192.168.1.1;
option domain-name-servers 8.8.8.8, 8.8.4.4;
default-lease-time 600;
max-lease-time 7200;
}
```

در این مثال، کلاینت‌ها می‌توانند آدرس‌های بین ۱۹۲،۱۶۸،۱،۱۰ تا ۱۹۲،۱۶۸،۱،۵۰ دریافت کنند و روتر پیش‌فرض ۱۹۲،۱۶۸،۱،۱ و سرورهای DNS گوگل برای نام‌گذاری دامنه‌ها ارائه شده است.

نرم‌افزار DHCP در لینوکس

امروزه می‌توانید انواع مختلفی از دستگاه‌ها (Devices) را در شبکه پیکربندی کنید تا به عنوان سرور DHCP (DHCP Server) عمل کنند. بیشتر روترها (Routers) این سرویس را ارائه می‌دهند و همچنین اکثر سیستم‌های عامل سرور (Server-Oriented Operating Systems) مانند ویندوز سرور (Windows Server) و البته سرورهای لینوکس نیز این قابلیت را دارند.

محبوب‌ترین بسته نرم‌افزاری سرور DHCP لینوکس توسط ISC – Internet Systems Consortium کنسرسیوم سیستم‌های اینترنتی (نگهداری می‌شود و به نام DHCPd شناخته می‌شود. تقریباً همه توزیع‌های سرور لینوکس این بسته را در مخازن نرم‌افزاری (Software Repositories) خود دارند، از جمله توزیع‌های مبتنی بر Red Hat و Debian.

زمانی که یک سرور DHCP روی شبکه خود راه‌اندازی کردید، باید به کلاینت‌های لینوکس (Linux Clients) خود بگویید که از این سرور برای دریافت آدرس‌های شبکه (Network Addresses) استفاده کنند. برای این کار به یک بسته نرم‌افزاری کلاینت DHCP (DHCP Client Software Package) نیاز دارید. سه بسته محبوب کلاینت DHCP برای لینوکس وجود دارد:

- dhcpcd
- pump
- dhclient

بیشتر توزیع‌های مبتنی بر Red Hat و Debian از بسته dhclient استفاده می‌کنند و حتی در برخی مواقع این بسته به طور پیش‌فرض نصب می‌شود، وقتی که یک کارت شبکه (Network Card) در حین فرآیند نصب سیستم شناسایی می‌شود. در اکثر محیط‌های دسکتاپ لینوکس، نیازی به انجام هیچ پیکربندی شبکه‌ای ندارید؛ برنامه نصب نرم‌افزار کلاینت DHCP را به طور خودکار نصب و فعال می‌کند. تنها کاری که باید انجام دهید این است که کابل شبکه را وصل کنید یا به شبکه بی‌سیم متصل شوید.

نصب سرور DHCP در لینوکس

اگر نیاز دارید یک سرور DHCP روی شبکه محلی خود پیاده‌سازی کنید، باید بسته ISC DHCPd را روی سرور لینوکس خود نصب کنید. برنامه DHCPd به اندازه‌ای محبوب است که در اکثر مخازن توزیع‌های لینوکس به طور پیش‌فرض موجود است، اما متأسفانه در توزیع‌های مختلف لینوکس با نام‌های متفاوتی شناخته می‌شود.

در حال حاضر، توزیع‌های مبتنی بر Debian این بسته را با نام isc-dhcp-server می‌شناسند (البته در نسخه‌های قدیمی‌تر، آن را dhcp3-server می‌نامیدند). برای نصب برنامه DHCPd، می‌توانید از ابزار استاندارد apt-get استفاده کنید:

```
$ sudo apt-get install isc-dhcp-server
```

برای توزیع‌های مبتنی بر Red Hat، نام بسته به سادگی dhcp است:

```
# yum install dhcp
```

PAM از ماژول‌ها (Modules) استفاده می‌کند تا درخواست احراز هویت از اپلیکیشن را به فرمت مناسب سیستم احراز هویت زیرین تبدیل کند و سپس پاسخ را به فرمت استاندارد قابل فهم برای اپلیکیشن‌ها تبدیل نماید.

مثال ماژول‌های PAM

کلید PAM ماژول‌های احراز هویت زیرین هستند. هر ماژول از یک روش احراز هویت پشتیبانی می‌کند، اما شما می‌توانید چندین ماژول را در یک نصب PAM نصب کنید تا از چندین روش احراز هویت پشتیبانی شود.

ماژول (Module)	توضیح (Description)
<code>pam_unix.so</code>	استفاده از فایل‌های استاندارد <code>/etc/passwd</code> و <code>/etc/shadow</code>
<code>pam_krb5.so</code>	استفاده از سیستم احراز هویت Kerberos 5
<code>pam_ldap.so</code>	استفاده از سرور Lightweight Directory Access Protocol – LDAP
<code>pam_nis.so</code>	استفاده از سرور NIS
<code>pam_sss.so</code>	استفاده از System Security Services Daemon – SSSD
<code>pam_userdb.so</code>	استفاده از فایل <code>db</code> استاندارد برای احراز هویت

علاوه بر ماژول‌های احراز هویت، PAM ماژول‌های کتابخانه‌ای اضافی (Library Modules) دارد که ویژگی‌های اضافی برای مدیریت حساب کاربری و سیاست‌های امنیتی ارائه می‌دهند، مانند:

ماژول (Module)	توضیح (Description)
<code>pam_access.so</code>	ارائه ورود ناشناس (Anonymous Logins) برای سرورهای FTP عمومی
<code>pam_chroot.so</code>	ایجاد یک محیط محدود (Locked Down Environment) برای ورود
<code>pam_console.so</code>	فراهم کردن محیط ورود کنسول (Console Login Environment)
<code>pam_cracklib.so</code>	بررسی رمز عبور (Password Strength)
<code>pam_deny.so</code>	جلوگیری از ورود به سیستم – (Login Attempts) معمولاً به عنوان پیش‌فرض
<code>pam_env.so</code>	تنظیم یا حذف متغیرهای محیطی (Environment Variables)
<code>pam_lastlog.so</code>	ارائه زمان آخرین ورود کاربر
<code>pam_limits.so</code>	اعمال محدودیت منابع (Resource Limits) مانند تعداد فایل‌های باز
<code>pam_listfile.so</code>	اجازه یا جلوگیری از دسترسی بر اساس فایل لیست (List File)

تقریباً همه توزیع‌های لینوکس PAM را به طور پیش‌فرض نصب و استفاده می‌کنند، حتی اگر فقط از فایل استاندارد `/etc/passwd` برای احراز هویت استفاده شود. نحوه استفاده هر اپلیکیشن از PAM توسط فایل پیکربندی PAM (PAM Configuration File) کنترل می‌شود.

پیکربندی PAM (Configuring PAM)

هر اپلیکیشنی (Application) که از PAM استفاده می‌کند، باید یک پیکربندی (Configuration Setup) داشته باشد که مشخص کند کدام ماژول احراز هویت (Authentication Module) باید استفاده شود و چگونه باید استفاده شود. متأسفانه، توزیع‌های مختلف لینوکس چندین روش متفاوت برای مدیریت سیستم پیکربندی PAM دارند.

روش اول: فایل واحد پیکربندی `/etc/pam.conf`

یکی از روش‌ها این است که همه پیکربندی‌های اپلیکیشن‌ها در یک فایل واحد (`/etc/pam.conf`) قرار داده شود. هر خط در این فایل دارای فرمت زیر است:

```
service type control module arguments
```

در پایگاه داده سلسله‌مراتبی، اشیاء به صورت درختی (Tree-Like) به یکدیگر متصل هستند. هر درخت LDAP باید یک شیء ریشه (Root Object) داشته باشد و اشیاء دیگر به ترتیب به آن متصل شوند. اشیاء از طریق ارجاع (Reference) به سایر اشیاء متصل می‌شوند، مشابه سیستم نام‌گذاری DNS.

- **مزیت** ساختار سلسله‌مراتبی: زمان خواندن داده‌ها سریع است، اما زمان نوشتن کند است. این روش فرض می‌کند که یک شیء پس از درج در پایگاه داده، چندین بار خوانده می‌شود اما به ندرت به‌روزرسانی می‌شود. این برای اکثر شبکه‌ها مناسب است، زیرا نام کاربری یکبار ایجاد می‌شود و سرورها هر بار که کاربر وارد می‌شود، آن را می‌خوانند.

اجزای پایگاه داده LDAP

پایگاه داده LDAP شامل یک یا چند اسکیمای (Schema) است. اسکیمای تعریف می‌کند که چه اشیایی (Objects) در پایگاه داده وجود دارند و چگونه به هم مرتبط هستند. هر شیء پایگاه داده توسط کلاس شیء (Object Class) تعریف شده و یک شناسه منحصر به فرد (Unique Object ID) به آن اختصاص داده می‌شود.

- کلاس شیء: یک قالب شامل مجموعه‌ای از ویژگی‌ها (Attributes) که شیء دارد.
- ویژگی‌ها: اطلاعات واقعی شیء را ذخیره می‌کنند، مانند نام کاربری برای شیء کاربر یا نام سرور برای شیء سرور. هر ویژگی شامل نام (Name) و مقدار داده مرتبط (Associated Data Value) است.

نامگذاری اشیاء در LDAP

در LDAP، درخت LDAP از استاندارد X.500 برای نامگذاری اشیاء استفاده می‌کند. مثال:

```
o="Sample, Inc.", c="US"
```

این نام به شیء شرکت **Sample, Inc.** که در ایالات متحده واقع شده اشاره دارد.

o و c به ویژگی‌های شیء اشاره دارند: سازمان (Organization) و کشور (Country).

برای ساده‌تر کردن نام‌ها، بسیاری از شرکت‌ها از نام **دامنه ثبت شده خود** برای درخت LDAP استفاده می‌کنند، به شکل زیر:

```
dc="sample", dc="com"
```

dc مخفف **Domain Component** است.

سپس می‌توان اشیاء اضافی زیر درخت شرکت ایجاد کرد. هر شیء باید با یک نام **منحصر به فرد (Distinguished Name - DN)** در پایگاه داده تعریف شود، مانند:

```
cn="rich", dc="engineering", dc="mycompany", dc="com"
```

این روش امکان ایجاد شاخه‌ها (Branches) و اشیاء نگهدارنده (Container Objects) را برای مدیریت بهتر پایگاه داده فراهم می‌کند.

فرمت تبادل داده LDAP (LDIF)

با توجه به چندین شیء و ویژگی‌ها، گاهی پیدا کردن داده مورد نظر دشوار است. **LDAP Data Interchange Format (LDIF)** یک استاندارد متنی برای تعریف اشیاء LDAP و ویژگی‌های آن‌ها است.

```
dn: <distinguished name>
```

```
<attrtype>: <attrvalue>
```

```
<attrtype>: <attrvalue>
```

```
...
```

contact me:

HosseinSeilani

Designer, Developer and Linux sysadmin

seilany.ir

learninghive.ir

emperor-os.ir

Hubuntu.ir

predator-os.ir

telegram: @seilany



Founder and Developer of Emperor-OS, Hubuntu and Predator-OS. I bring significant experience as a Linux/Windows sysadmin and graphical web design, UX/UI to the Open Source Community.



Emperor-OS



Predator-OS