

Course Introduction

Linux System Administration
Fall 2023

Course Goals

- To help students gain an understanding of how to manage a UNIX or Linux environment, and provide a marketable set of technical skills.
 - Installing and configuring UNIX/Linux Systems
 - Routine maintenance
 - Security
 - **Troubleshooting**

Course Objectives

- Demonstrate proficiency with the Linux command line, filesystem, and core components of UNIX
- Effectively operate a Linux system inside of a network environment
- Design and deploy Linux systems to accomplish tasks such as: serving web content, sharing files, firewalls, etc.

Grading

Assignment	Fraction
Labs	50%
Midterm	20%
Final	20%
Participation	10%

- Some assignments will have extra credit
- Assignments have extra tasks for grad students

Letter	Numerical
A	95-100
A-	90-94.99
B+	85-89.99
B	80-84.99
B-	75-79.99
C+	70-74.99
C	65-69.99
C-	60-64.99
D+	55-59.99
D	50-54.99
D-	40-49.99
F	0-39.99

Class Attendance

- For the in person section, attendance is expected
 - If you are sick, attend the remote section
- For the remote section, stay current
 - The lecture expands on expectations for labs
 - Lab solutions are discussed in class

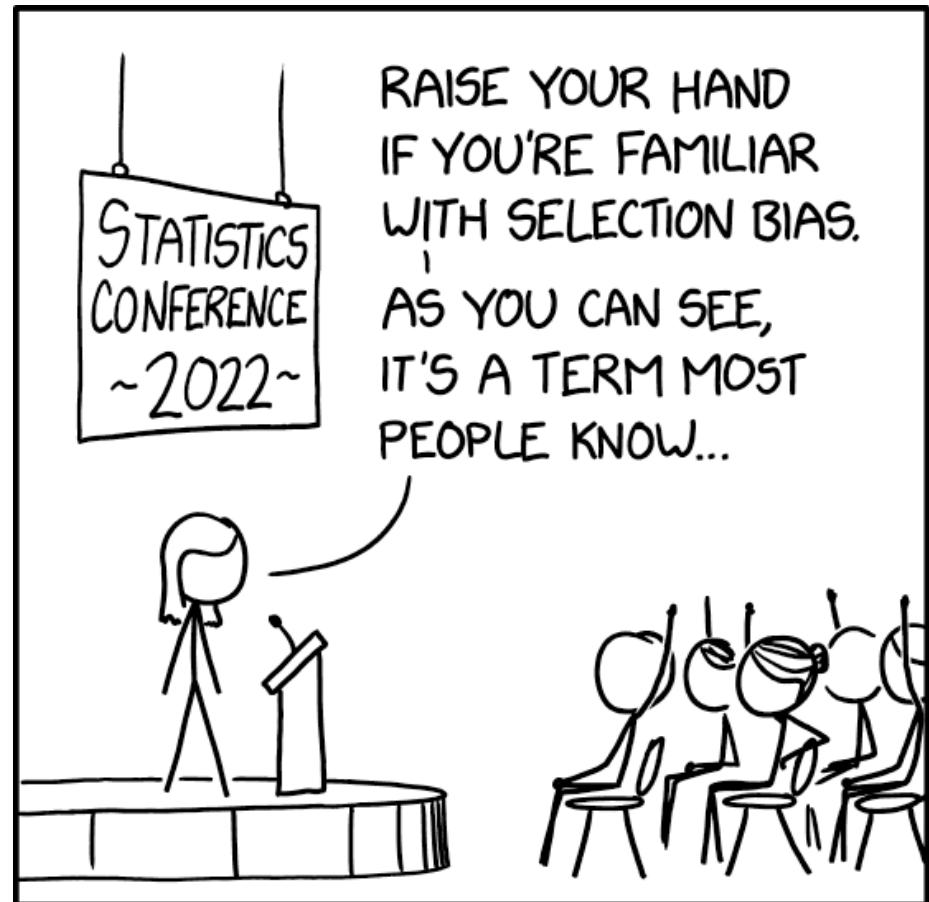
Field Trip

- Each semester we go to touch some hardware
- Supercomputing 2023 (SC23)
 - Colorado Convention Center Nov 12-17, 2023
 - Student Tour Wednesday Nov 15

My challenge

- System administration cannot be learned in a linear fashion
 - Some things may seem repetitive
- I discover commands all the time
- *TIMTOWTDI*

(You should read the jokes on the slides)



Final Exam

- The final is a practical exam
 - Open access to textbook, notes and internet, but NO help from other people or ChatGPT
- Timed task
 - Recover hung machines or services
 - Very difficult unless you do the labs yourself

Deadlines

- Most labs are due at 23:59 on Monday.
- No penalty grace period until 08:00 on Tuesday
 - *If you leave this until Monday evening, it will typically not turn out well*
- Extensions can be granted if requested **before** the deadline, but cannot be later than the start of the next class period

Grading

- Grading is done using a grading script that verifies that all the requirements are satisfied
 - The script is **very** picky about exact naming of scripts, user and group names, directories, etc.
 - Watch for adding extensions (e.g. .py or .sh - that's a Windows thing - Linux uses a # ! line)
- *An important part of the system administration is to develop your own test and verification scripts*

Textbook

- **Unix and Linux System Administration Handbook 5ed**
 - Nemeth, Snyder, Hein, Mackin & Whaley
 - ISBN-13: 978-0134277554
- Required for readings.

About Evi

- Evi Nemeth
 - 6/7/1940 – 6//?/2013
 - CU Boulder 1980-2001
- Lead author of textbook
- Pioneer in establishing system administration as a field of study

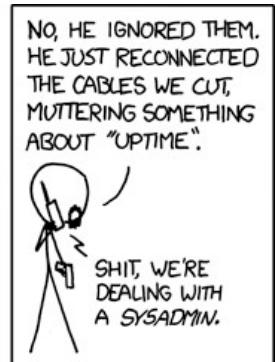
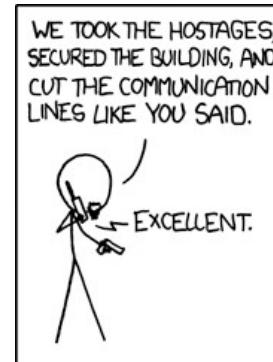


Contact Information

- Willem A (Vlakkies) Schreüder
 - vlakkies@colorado.edu
 - ECOT 732
 - Piazza
- Office Hours
 - Monday 3-4pm by Zoom
 - Thursday 11am-noon in person or by Zoom
 - Email me to meet by Zoom or in person at other times

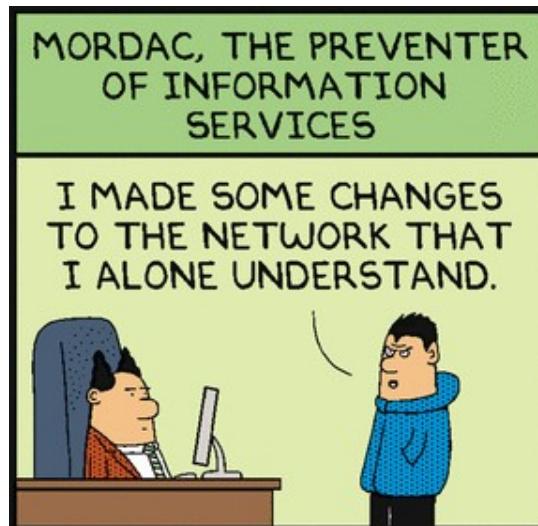
System Administration Principles

- As shown when using sudo the first time:
 - Respect the privacy of others.
 - Think before you type.
 - With great power comes great responsibility.
- Our Motto: D.F.I.U.
 - Don't Mess It Up



Don't be this guy

- Never make changes shortly before leaving
 - *Do not try to do the homework an hour before the deadline*
- Reboot the system after touching critical configs



System administration tasks

- Design new systems
- Provision hardware
- Add/remove hardware
- Install/upgrade software
- Provision users
- Perform backups
- Monitor systems
- Document systems
- Monitor security
- Support operations

UNIX Principles

- *UNIX is simple. It just takes a genius to understand its simplicity* -Dennis Ritchie
- Everything is a file
 - Files are streams of bytes
 - Programs consume and produce text streams
 - Commands are executable files
 - #! sets interpreter for scripts

UNIX Philosophy (Doug McIlroy)

- Write programs that do one thing and do it well.
- Write programs to work together.
- Write programs to handle text streams, because that is a universal interface.

A brief history of Linux

- 1964 MULTICS (MIT, GE, AT&T/Bell Labs)
- 1969 Unics/UNIX (Bell labs)
- 1983 GNU (Richard M. Stallman)
- 1987 MINIX (Tanenbaum/Vrije Universiteit)
- 1991 Linux (Linus Torvalds)

MULTICS (MIT,GE,AT&T/Bell Labs)

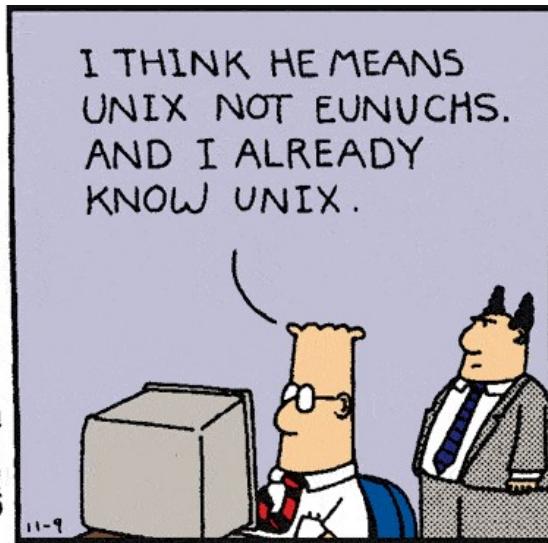
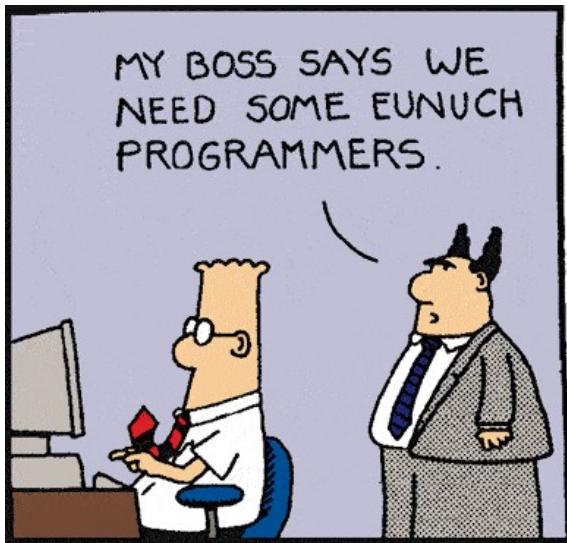
Multiplexed Information and Computing Service



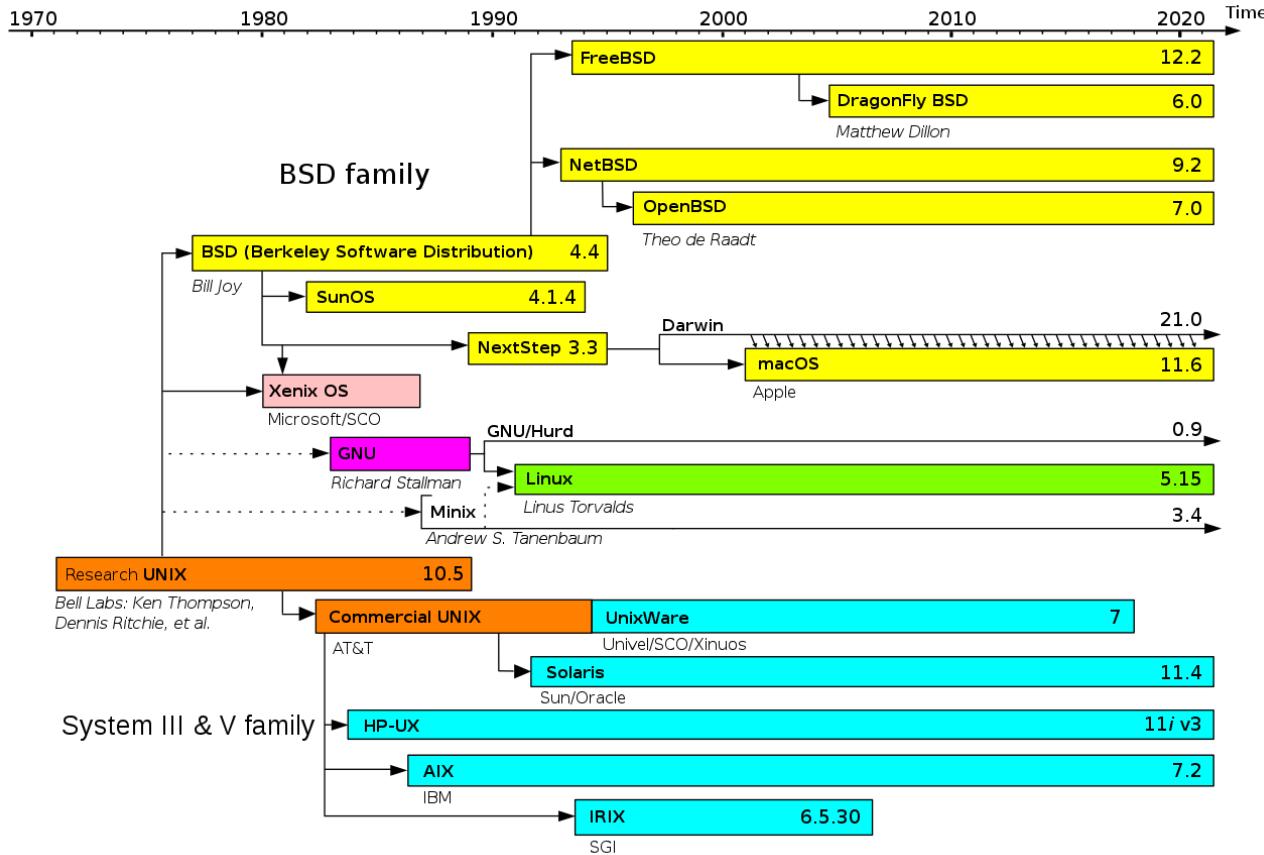
- Unics (Bell labs)
Uniplexed Information and Computing Service



Unics, UNIX, Unix or *nix?



UNIX family



GNU (Gnu is Not Unix)



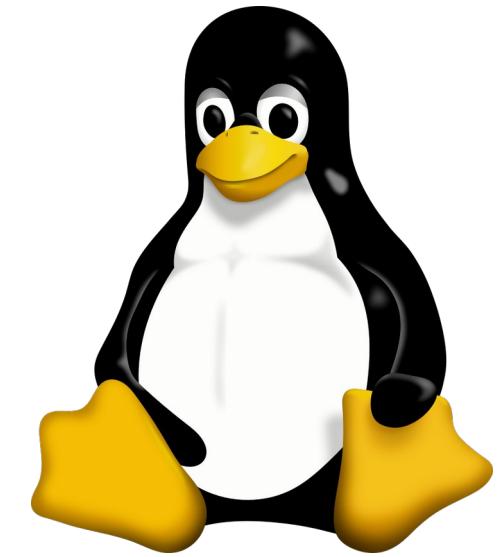
- Richard M Stallman (RMS) announced plans to create an **entirely free** UNIX like operating system in 1983
 - Free as in speech, not free as in beer
 - GNU Public License (GPL)
- Most of the core components, except for the kernel, were completed by 1987 (GNU hurd)
 - gcc, g++ and many UNIX utilities often used on commercial UNIX platforms like HP-UX and SunOS

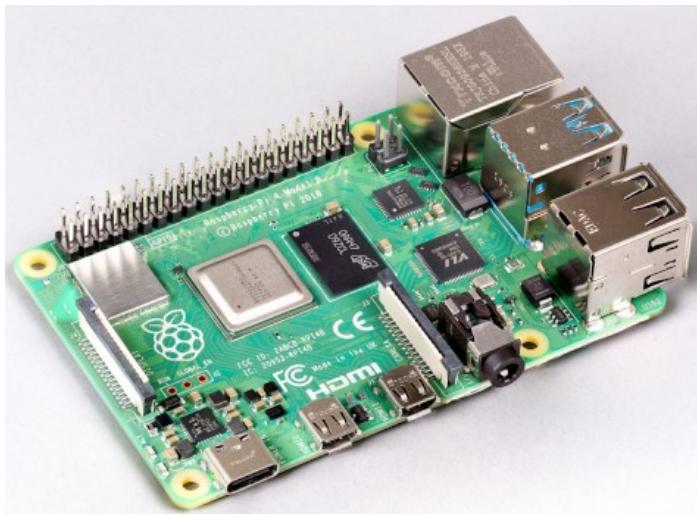
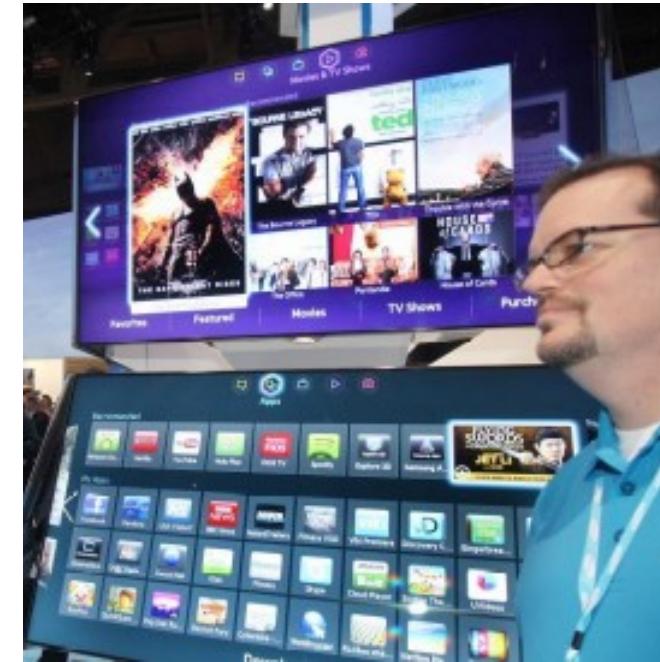
MINIX (mini-Unix)

- Microkernel implementation on IBM PC
 - call compatible with UNIX V7
 - Only about 12,000 lines of C
- Used to teach operating systems
 - Operating Systems: Design and Implementation
Andrew S Tanenbaum, Vrije Universiteit
- Not free, but source code available
- Inspiration for Linux

And then there was Linux

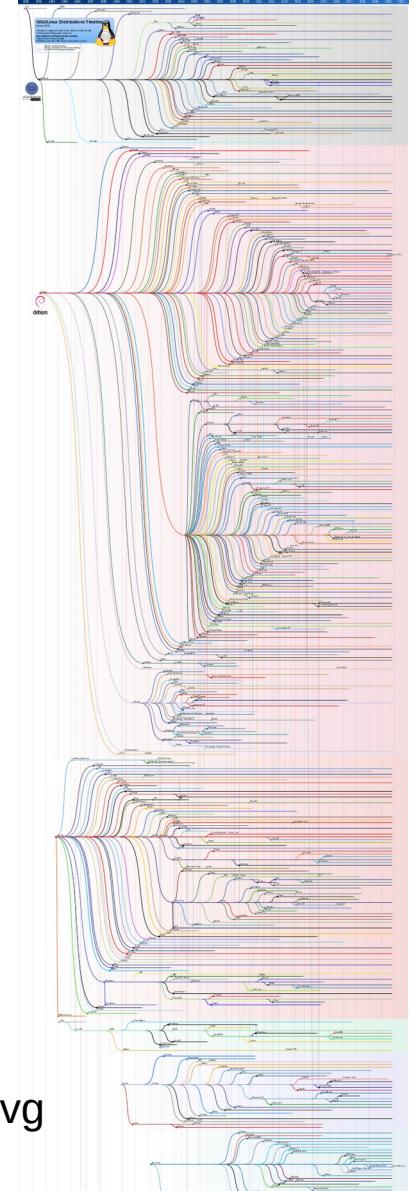
- Linus Torvalds (1991)
 - New kernel released under GPL
 - GNU utilities to complete OS (GNU/Linux)
- Quickly adopted from desktop to Big Iron to IoT
 - Distributions provide installer and package manager
 - Initially CDs, Internet make downloads possible
- Desktop matured with KDE & GNOME (late 90s)
 - Android largest installed base of any OS





Linux Distributions

- Slackware
 - SuSE
- Debian
 - Ubuntu
- Red Hat
 - CentOS



https://upload.wikimedia.org/wikipedia/commons/1/1b/Linux_Distribution_Timeline.svg

Choosing a Distribution

- debian
 - Stresses free software
 - Base for Ubuntu, Raspbian, etc
 - Upgrades across major revisions
 - Fairly quick updating of packages to latest
- RedHat
 - Stresses stability
 - Commercial product
 - CentOS, Rocky, Alma are free versions
 - Major revision upgrades are often new installs
 - Not bleeding edge

Berkeley Software Distribution

- Early derivative of Research Unix
 - University of California Berkeley
- Pioneered networking
- Permissive license
 - Basis for commercial products
 - SunOS, macOS
- FreeBSD
 - Stresses ports
- OpenBSD
 - Stresses security
- NetBSD
 - Stresses hardware support
- Many others

Lab 1

- Access the CS vSphere machines
- Change the root password on all machines
- Install `vim` on all machines
- Grad Students: Also do FreeBSD

Shell Basics

Linux System Administration
Fall 2023

The command line

- You need to know your way around the CLI
 - When stuff breaks that may be all you have
 - Readily runs remotely via ssh
 - One liners can do complex things
- Scripting simplifies your life
 - Creates new custom commands
 - Laziness can be a virtue

System access basics for Lab 1

- root can do *anything*
 - Root logins can be restricted
- User access by
 - UID - user id
 - GID - group id
 - users can belong to multiple groups
- File permissions control access to commands
- sudo - run as root
 - access by user or group
- Administering users
 - useradd
 - userdel
 - usermod
- Figure out how to allow privileged access to your non-root login

The Unix ABCs

- A is for awk, which runs like a snail
- B is for biff, which reads all your mail
- C is for cc, as hackers recall
- D is for dd, the command that does all
- E is for emacs, which rebinds your keys
- F is for fsck, which rebuilds your trees
- G is for grep, a clever detective
- H is for halt, which may seem defective
- I is for indent, which rarely amuses
- J is for join, which nobody uses
- K is for kill, which makes you the boss
- L is for lex, which is missing from DOS
- M is for more, from which less was begot
- N is for nice, which really is not
- O is for od, which prints out things nice
- P is for passwd, which reads in strings twice
- Q is for quota, a Berkeley-type fable
- R is for ranlib, for sorting a table
- S is for spell, which attempts to belittle
- T is for true, which does very little
- U is for uniq, which is used after sort
- V is for vi, which is hard to abort
- W is for whoami, which tells you your name
- X is, well, X, of dubious fame
- Y is for yes, which makes an impression, and
- Z is for zcat, which handles compression.

Figuring stuff out

- man - online help
 - man sections
 - 1 User Commands
 - 2 System Calls
 - 3 C Library Functions
 - 4 Devices and Special Files
 - 5 File Formats and Conventions
- apropos (man -k)
 - and the SEE ALSO part of man pages
- -h flag on many commands

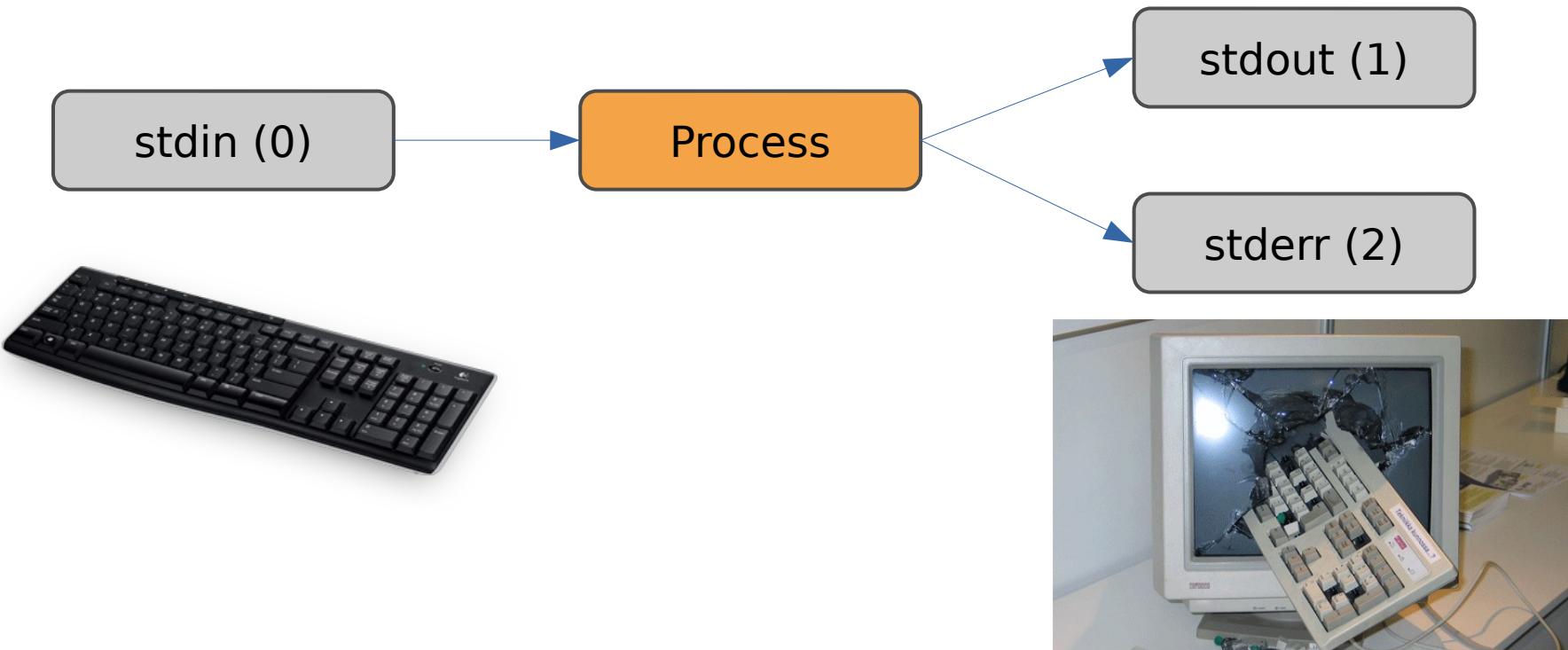
Shells

- Thompson Shell (1971)
- sh Bourne Shell (1977)
- csh C Shell (1978)
- ksh Korn Shell (1983)
- tcsh T Shell (1983)
- bash Bourne-again Shell (1989)

Anatomy of a shell

- `user@host:dir# command arg1 arg2 arg3`
 - Prompt (PS1)
 - Command expanded by shell
 - Wildcards and variables
- ; or \n separates commands
- Whitespace separates arguments
- Data can be redirected with <, > and |

Unix process (commands)



Redirecting stdout

- > Write output to file
- >> Append output to file
- | Pipe output to next program

Redirecting stdin

< Read input from file

```
mail vlakkies@colorado.edu < message.txt
```

<< Read input until tag

```
mail vlakkies@colorado.edu << EOF
This is the text
of this email.
EOF
```

<<< Read input from current line

```
mail vlakkies@colorado.edu <<< 'This is the text'
mail vlakkies@colorado.edu <<< "Mail from $PWD"
```

Special Redirection

2> Redirect stderr

2>&1 Merge stderr into stdout

Converting stdio to arguments

- xargs
 - Change stdin to command line arguments
- grep -l -dskip foo * | xargs sed -i s/foo/bar/
 - grep
 - -l only list file named
 - -dskip ignore directories
 - sed
 - change foo to bar in place (-i)

curses (ncurses)

- Text-based API used by CLI programs
 - top, screen, vim, cping, Midnight Commander, ...
- ncurses (and other) re-implementation
- Works very well over ssh connections
- Uses terminfo to get terminal capabilities

Variables

- Set syntax depends on shell
 - csh: set x=1
 - bash: x=1
- Usage
 - echo "x=\$x"
- Promoting to global
 - export x

Quotes

- Single quote (') - no variable expansion
 - echo 'This string is dollar x: \$x'
- Double quote ("") - expand variables
 - echo "dollar x equals \$x"
- Back quote (`) - execute as command
 - vi `grep -l 'Das Foo' *.txt`

Editors

- sed - streaming editor
- vi - visual editor
 - vim - Vi IMproved
- nano
- emacs

Stuff to know

- PATH Where to find commands
 - which *command*
- Special directories: . .. ~
- Dot files and directories
- Example filenames: foo bar
- Sourcing files: . script

Commonly used commands

- ls - list
- cp - copy
- mv - move
- rm - remove
- directory commands
 - cd - change
 - mkdir - make
 - rmdir - remove
- echo - write to screen
- cat - concatenate
- more - paginate
- grep - search
- xargs - convert to arguments
- ssh - remote shell
- scp - remote copy
- rsync - remote sync

Process management

- Ctrl-C terminate the current task
- Ctrl-Z suspend the current task
- bg resume suspended task in the background
- fg resume task in foreground

Commands for working with text

- `wc` - word/line counts
- `sort` - ordering
- `uniq` - repeated lines
- `cut` - show columns
- `paste` - merge files
- `join` - merge on field
- Streaming editors
 - `sed`
 - `awk`
 - `tr`
- Comparing files
 - `diff`
 - `sdiff`
 - `cmp`

System Commands 1

- Processes
 - who
 - ps
 - top
 - kill
 - renice
- Networking
 - ip addr
 - ip route
 - ping
 - traceroute
 - dig
 - nslookup

System commands 2

- File system
 - fsck
 - mount
 - umount
 - fdisk
 - mkfs
 - df
 - du
 - sync
- File manipulation
 - chmod
 - chown
 - touch
 - truncate
- Compression
 - gzip, gunzip
 - bzip2, bunzip2
 - zcat, zgrep, zmore, zdiff
- Swiss army chainsaw copier
 - dd

Document management

- LibreOffice (WYSIWYG)
 - `lowriter`, `localc`, `loimpress`, `lodraw`, ...
- Typesetting
 - `troff`, `nroff`, `groff`, ...
 - `TeX`, `LaTeX`, `pdfTeX`, `pdfLaTeX`, ...

Unix Command Line Basics

Linux System Administration
Fall 2023

bash command line

- Default shell on modern Linux
 - Editing, history, autocompletion
- Built-in commands
 - if, for, while, case, select, alias, ...
- Parameter manipulation
- Shell variables
- Compound with | and ;

Bash startup Files

https://www.gnu.org/software/bash/manual/html_node/Bash-Startup-Files.html

- Interactive login
 - /etc/profile
 - First existing from
 - ~/.bash_profile
 - ~/.bash_login
 - ~/.profile
- Interactive non-login shell
 - ~/.bashrc
- Non-interactively
 - Defined by \$BASH_ENV
- Interactive login should explicitly source ~/.bashrc in profile
- Profile typically sets \$PATH and umask
- ~/.bashrc sets most things
 - prompt (\$PS1)
 - terminal (\$TERM)
 - aliases (~/.bash_aliases)
 - other variables as needed

ssh (secure shell)

- replaces telnet & rlogin/rsh
 - encrypted connection
 - basis for scp, rsync, etc
- Our go-to tool for access
 - Major goal of exams
 - ssh tunnel (-L and -R)
- Password-less login
 - `~/.ssh/authorized_keys`
- `ssh user@host`
- Server side sshd
- Default port 22
 - `p` for other port
- `ssh user@host 'command'`
- Forward X11 with `-X` or `-Y`
- Debug with `-vvvv`
- root logins may be disabled

Editing the command line

- Left and Right arrow, Home and End moves cursor
- Backspace and delete
- Tab completes file name
 - Can be context sensitive
- Ctrl-C aborts editing
- Up and down arrows scroll through the command history
- Ctrl-R searches command history for pattern
 - Ctrl-R searches for next
 - ESC to stop and edit
 - ENTER to stop and execute

Script arguments

- Positional parameters
 - `$0` name of script
 - `$n` parameter *n*
 - `$*` all parameters from 1
- Wildcard expansion is done by the shell
 - Script sees expanded
- Useful shell parameters
 - `$EUID` effective UID
 - 0 when run by sudo
 - `$PWD` current directory
 - `$HOSTNAME` host
 - `$LINES` window height
 - `$COLUMNS` window width

Compound commands

- Unix approach: commands should do one thing well.
Compound commands allow complex behavior
- Separate with ;

```
for file in *.txt; do mv $file $file.0; done
```
- Pipeline with |

```
awk -F: '{print $7}' /etc/passwd|sort|uniq -c
```
- Convert with xargs

```
grep -l -dskip foo *|xargs sed -i s/foo/bar/
```

One liners

- **Login shells in /etc/passwd**

```
awk -F: '{print $7}' /etc/passwd | sort | uniq -c
```

- **Do something on multiple hosts**

```
for host in foo bar; do ssh $host dnf install mlocate; done
```

- **Do something on specific files**

```
file * | grep CRLF | sed 's/: .*//' | xargs -l dos2unix
```

sudo

- Run command as root
 - Training wheels for root
 - Can limit commands
 - Provides audit trail
- /etc/sudoers
 - Group **sudo** or **wheel**
- ***USE IT!!!***
 - D.F.I.U.

MAKE ME A SANDWICH.



WHAT? MAKE
IT YOURSELF.

SUDO MAKE ME
A SANDWICH.



/
OKAY.



<https://xkcd.com/149/>

Files and directories

- Everything is a file
- Unified directory tree
 - / is the directory separator
 - file systems mount as directories
- File names are arbitrary strings
 - White space in file names are a nightmare
 - Extensions are not definitive

Special file types

- directory - list of file names and inode pointers
 - hard link - directory entry to existing inode
- symbolic link - path to another file
- pseudo files
 - FIFO (named pipe)
 - socket
 - device file (block or character)

Where are stuff?

/boot – boot loader files

/etc – system configuration

/bin – essential binaries (`ls`, `cat`, `cp`, `mv`)

/sbin – essential system binaries (`mount`, `mkfs`, `fsck`, `modprobe`)

/usr – user utilities and files (`/usr/bin`, `/usr/share`, `/usr/lib`)

/var – variable files (`/var/log`, `/var/mail`, `/var/spool`)

/tmp – temporary files (cleaned out on boot)

/dev – device files

/proc – virtual system files (kernel access)

/home – user data

Manipulating the file system

- Directories
 - cd - change
 - ls - list
 - pwd - show current
 - mkdir - create new
 - rmdir - remove
- Files
 - cp - copy
 - mv - move
 - rm - remove
 - ln - link
 - cat - concatenate
 - scp - remote copy
 - rsync -remote sync

ls flags

- l long listing
- a also show dot files
- h show human format
- t sort by time
- S sort by size
- d show directory
- i show inode
- r reverse
- 1 list one file per line

- By default, files starting with . (period) are not shown
- List file by type
 - color color code
 - F append */=>@|
- Show files in human readable format ordered by time in reverse
 - ls -ltrh

Special directory names

- / root directory
- . current directory \$PWD
- .. parent directory
- ~ home directory \$HOME
- previous directory on cd \$OLDPWD

Moving files locally

- Copy file(s)

`cp from to`

- i prompt on overwrite
- a preserve attributes
- r recursive

Copy multiple files if
to is a directory

- Move file or directory

`mv from to`

- i prompt on overwrite

Preserves attributes by default

Move multiple files if
to is a directory

Copying files remotely

- `scp from to`
 - p preserve attributes
 - r recursive
 - P port
- `rsync from to`
 - a preserve attributes
 - r recursive
 - delete-excluded
- Layered on ssh
- Specify remote as
`user@host:dir`
- rsync copies only when changed files very efficiently

Locating files

- `find dirs`
 - Search directories
 - `name pattern`
 - `iname pattern`
 - Lots of options to find by type, time, permissions, etc.
- `which command`
 - Uses \$PATH to find executable for `command`
- `locate pattern`
 - Not installed by default
 - Database of files
 - `updatedb` scans filesystems

Finding out more about files

- `file`
 - Determines file properties by examining the contents of the file
 - Extensions are not definitive
- `stat`
 - Shows details of the file returned by `stat()` system call
 - Permissions
 - Disk usage
 - Times

Comparing files

- `diff file1 file2`
 - compare two files
- `diff -r dir1 dir2`
 - compare two trees
- `sdiff file1 file2`
 - show differences side-by-side
- `zdiff file1 file2.gz`
 - compare gzipped files

Useful diff flags

- q only report that files differ
- i ignore case
- b ignore changes in the amount of whitespace
- w ignore all white space
- Z ignore trailing whitespace
- B ignore blank lines

File permissions

- Owner:Group:Other
 - r - read (4)
 - w - write (2)
 - x - execute (1)
 - Directory x=list
- Example
 - `rwxr-xr--`
- `ls -l`
 - show permissions
- `chmod`
 - `chmod ug=rwx`
 - `chmod 775`
 - Values are octal
 - `chmod g+w`

Showing what is in a file

- cat – show all
- more - paginate
- head – first n lines
- tail – last n lines
 - f to follow changes
- grep pattern file(s)
 - lines with pattern
 - v to invert selection
- awk action file(s)
 - Show users in passwd
`awk -F: '{print $1}' /etc/passwd`

aliases

- Creates new or modifies commands
- Applies when it is the first word of a simple command
- Expanded by bash
- `~/.bash_aliases`
- `alias cp='cp -i'`
- `alias mv='mv -i'`
- `alias rm='rm -i'`
- `alias ll='ls -l'`
- `alias vi='vim'`
- `alias -p`
- `unalias ls`

Bash if and for

```
for x in *
do
  if [ -f "$x" ]
  then
    echo "$x is a file"
  else
    echo "$x is something else"
  fi
done
```

- Loop over all files
- If it is of type file
 - say it is a file
- else
 - say it is something else

if

Conditional action

```
if condition  
then  
  commands  
fi
```

Alternative actions

```
if condition  
then  
  commands  
else  
  commands  
fi
```

Nested decisions

```
if condition  
then  
  commands  
  if condition  
  then  
    commands  
  else  
    commands  
  fi  
else  
  commands  
fi
```

Cascading decisions

```
if condition  
then  
  commands  
elif condition  
then  
  commands  
else  
  commands  
fi
```

Conditional expressions

- `if [$x == 0]`
 - Is the string '0'
 - `if [$x -gt 0]`
 - Is \$x positive
 - `if ["$x" != abc]`
 - Is the string not abc
 - `if [-z "$x"]`
 - Is this a blank string
 - `if [-r foo.txt]`
 - Is foo.txt readable
- Hints
 - Beware string vs. numeric comparisons
 - Bash defaults to strings
 - Whitespace matters
 - Quote strings that may be blank
 - 0 means true
 - Lots more on man page
 - CONDITIONAL EXPRESSIONS

case

```
case $variable in
    pattern1)
        commands;;
    pattern2)
        commands;;
    patternN)
        commands;;
    *)
        commands;;
esac
```

- Hints
 - Comparison is based on string matching
 - Wildcards
 - * matches anything
 - Clauses are done in order
 - First match wins
 - No action if none match
 - Can be a convenient if

Iteration

- **for** var **in** list; **do** commands; **done**
 - Execute commands for values in list
- **while** cmd1; **do** cmd2; **done**
 - Do cmd2 while cmd1 returns true
- **until** cmd1; **do** cmd2; **done**
 - Do cmd2 until cmd1 returns true

Return values

- Generally 0 means everything is OK
 - This is just like most system function calls
- Return (exit) value of the last command is \$?
- 0 is true in bash
- Example: Can we ping 8.8.8.8?

```
ping -c 1 8.8.8.8
if [ $? ]; then
    echo "8.8.8.8 pings OK"
fi
```

Whitespace Matters

- Assignment

date=3 vs. date = 3

- Expressions

if [\$?];then vs. if [\$?];then

- Range

for x in {1..5} vs. for x in {1 .. 5}

Quotes matter

```
x=`date`  
for i in {1..3}  
do  
    echo "$X"  
    sleep 1  
done
```

```
X="date"  
for i in {1..3}  
do  
    echo `\$X`  
    sleep 1  
done
```

```
x=`date`  
for i in {1..3}  
do  
    echo '$X'  
    sleep 1  
done
```

```
X='date'  
for i in {1..3}  
do  
    echo `\$X`  
    sleep 1  
done
```

Bash arithmetic

```
x=5
```

```
y=6
```

```
z=$x+$y
```

```
echo $z
```

```
5+6
```

```
((z=$x+$y))
```

```
echo $z
```

```
11
```

- Hints:

- Bash treats variables as strings
- Double parentheses means it is an arithmetic expressions
 - ((z=\$x+\$y))
 - z=\$((\$x+\$y))
- Result is a string to bash

Scripts

- Comment start with #
- Initial line #! invokes an interpreter
 - #!/bin/bash - bash
 - #!/usr/bin/python3 - python 3
- File must be executable
- **You should get good at bash and another scripting language like Python or Perl**

bash arrays

- Initialize

```
days=( 'M' 'W' 'F' )
```

- Assign

```
days[2]='Saturday'
```

- Append

```
days+= 'Sunday'
```

- Access one element

```
echo ${days[1]}
```

- Access all elements

```
echo ${days[*]}
```

```
echo ${days[@]}
```

- Number of elements

```
echo ${#days[*]}
```

- Index of elements

```
echo ${!days[*]} }
```

- Declaration

```
declare -a days
```

- Associative arrays

```
declare -A name
```

Bash functions

- Definition

```
myfunc () { commands; }
```

or

```
function myfunc { commands; }
```

- Arguments are by position

- \$1 \$2 \$3

- Return value in \$?

- Use local to make variables local to function

Parameter expansion

- Isolate variable
 - `${variable}`
 - `dir/${file}a.txt`
- Substring
 - `${name:off:len}`
- Substitution
 - `FILE=foo.txt`
 - `${FILE/txt/pdf} = foo.pdf`
- Remove prefix
 - `${parameter#word}`
- Remove suffix
 - `${parameter%word}`
- Set default value
 - `${parameter:-value}`
- Parameter length
 - `${#parameter}`

Basic networking

- `ifconfig` **configure interfaces**
- `route` **configure routing**
- `ip` **configure networking**
- `ping` **test connectivity**
- `traceroute` **test routing**
- `dig` **test DNS**
- `nslookup` **test DNS**

Downloading files

- Get data from a web or ftp server
 - ftp supports wildcards
- wget
 - m is great for mirroring entire directory tree
- curl (cURL)
 - Supports many more protocols

Users and Groups

Linux System Administration
Fall 2023

What is a User Account?

- A user account is a set of credentials that give a person access to the machine.
- There are several attributes associated with each account on a system.

Attributes of a User Account

- Username
- Password (password placeholder)
- User ID (UID)
- Primary Group ID (GID)
- Textual Description (Gecos)
- Home Directory
- Default Shell

What is a Group

- A group is a set of users that allows permissions to be granted in a more organized way.
 - File access
 - Used directly by some programs like sudo

Group Attributes

- Group Name
- Group ID (GID)
- Group Members

Special accounts

- System processes (daemons) need accounts on the machine too and you will sometimes see accounts for specific programs
 - Limit elevated access as much as possible
- What is running on the system

```
ps -eo user|sort|uniq -c
```

```
ps -eo user,pid,comm | sort
```

In the beginning

- User information stored in files
 - /etc/passwd
 - /etc/group
- Password encrypted but vulnerable to attack
- All credentials local

Hiding secrets

- /etc/passwd
 - User name, UID, etc
- /etc/shadow
 - Encrypted password, password aging

/etc/passwd format

mscott:x:5001:5000:Michael Scott:/home/mscott:/bin/bash

mscott - user name

x - password is encrypted and stored in /etc/shadow

5001 - User ID

5000 - Group Id

Michael Scott - Full name (GECOS)

/home/mscott - home directory

/bin/bash - login shell

/etc/shadow format

mscott:\$6\$hyz\$bx0cpZOlf0:18056:0:99999:7::

mscott - user name

\$6\$hyz.... encrypted password

- \$x\$ sets encryption algorithm
- First few digits is the salt

18056 - Last date password changed

0 - Minimum password age

99999 - Maximum days password is valid

7 - Days of warning before expiration

Inactive and Expire is blank

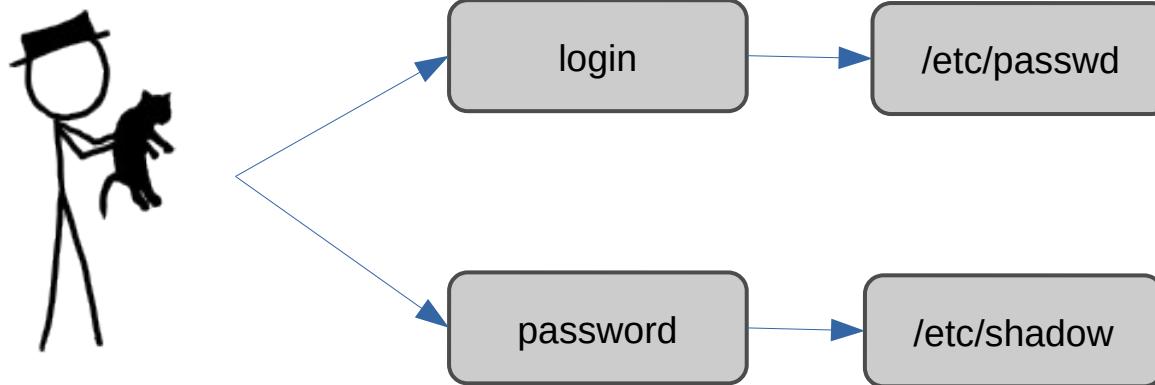
/etc/group format

managers:x:6001:mscott,jhalpert,dschrute

- managers - group name
- x - password in /etc/gshadow
- 6001 - Group ID
- mscott, jhalpert, dschrute - members
 - Users with the GID as their primary group will also be members of this group

Basic login

- User credentials checked against files



Networked User Information

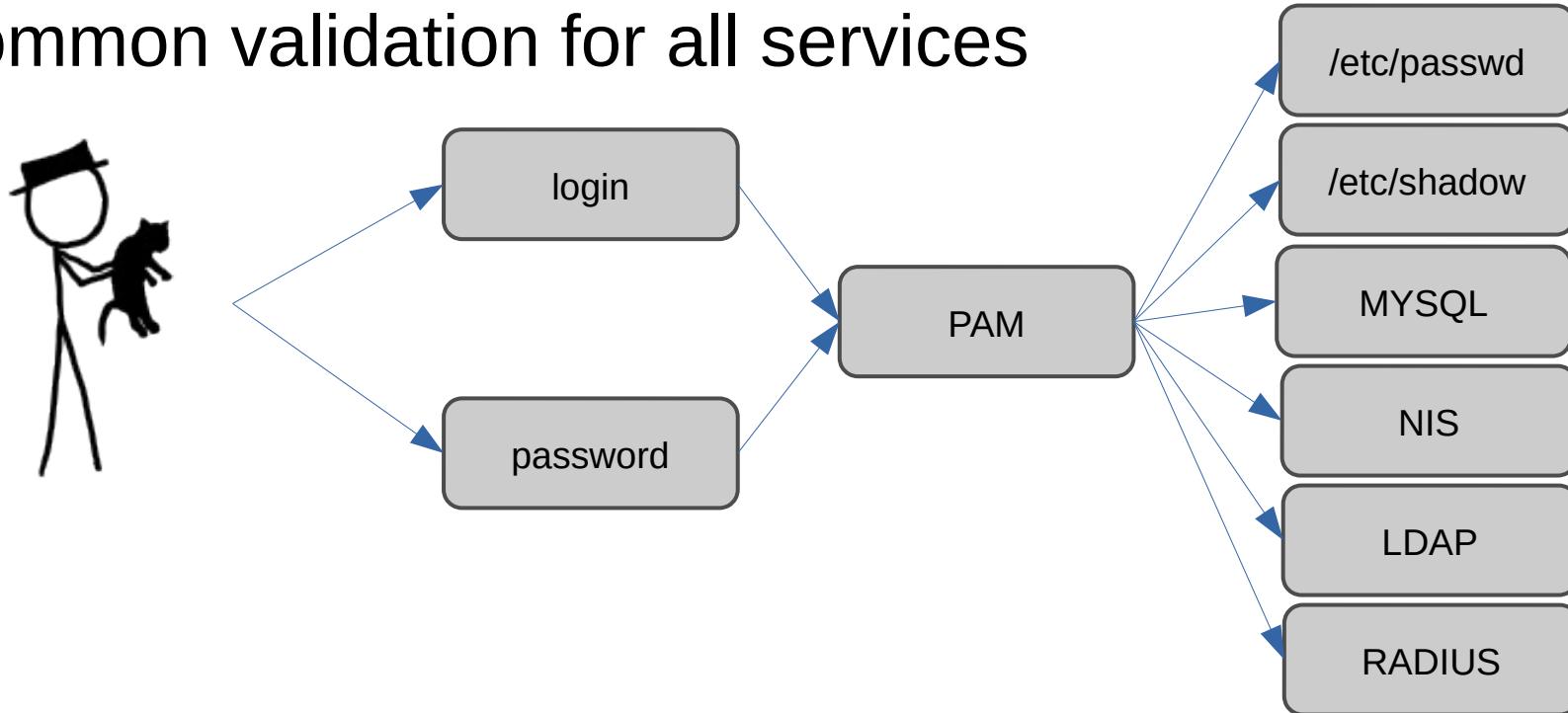
- NIS - Network Information Service
 - Originally called Yellow Pages
- LDAP - Lightweight Directory Access Protocol
- Kerberos
- many others

Which login is used?

- NSS - Name Service Switch
 - Sets order of evaluation for GNU C library
- PAM - Pluggable Authentication Module
 - Fine grained control of authentication

PAM login

- Pluggable Authentication Modules
- Common validation for all services



Adding a user

- `vipw`
 - Trusted vi /etc/passwd
 - Also -p
 - `-s` to edit /etc/shadow
 - `-g` to edit /etc/group
 - Also `vigr`
- `useradd login`
 - `u` - set userid
 - `U` - add private group
 - `g` - set GID or group
 - `G` - add groups
- Must set password
- Template for files in home directory in /etc/skel

passwd

- Allows the user to update their own password
- `passwd login`
 - Allows root to set the password for a user
- `chpasswd`
 - Change password in batch mode

groupadd

- Creates a new group
- -g sets GID
- -p can set a group password
- `groupadd -g 6001 managers`

adduser

- debian
 - Like useradd, but also prompts for name, GECOS and password
 - `adduser login group`
 - Adds *login* to *group*
- RedHat
 - Basically just useradd
- BSD
 - Detailed prompts

Removing a user and group

- `userdel login`
 - Removes the user
 - `-r` also remove home directory and mail spool
- `groupdel group`
 - Removes the group

`usermod` *login*

- Modifies the user attributes
 - l change login name
 - u change UID (also modifies mailbox and files in /home)
 - L lock (disable) login by removing password
 - g set primary group for user
 - G add user to this secondary group
 - p change password
 - s change shell

Change user info and shell.

Users can also change their own

- chfn
 - Change “finger” data
 - Full Name
 - GECOS data
 - Office
 - Phone number
- chsh
 - Change login shell for user
 - -l list available shells

Special groups

- wheel - traditional users with root access
 - Redhat: users with unlimited sudo
 - debian: wheel group renamed to sudo
- dialout - users with access to serial ports
- Private groups

Gotchas

- Group membership is updated on login
 - Log out or launch new login shell

File Permissions

Linux System Administration
Fall 2023

UNIX File Modes

- A unix file's mode is a collection of numbers that represents permissions.
 - Fundamental controls user's access to system.
 - r - Read controls data that can be read
 - w - Write controls what data can be written
 - x - Execute controls what commands can be run

Unix file modes

Permission	Symbolic	Binary	Octal
everything	rwx	111	7
read and write	rw-	110	6
read and execute	r-x	101	5
read only	r--	100	4
write and execute	-wx	011	3
write only	-w-	010	2
execute only	--x	001	1
none	---	000	0

Access Modes

- File
 - **r** can read file
 - **w** can modify file
 - **x** can execute file
 - run file as command
- Directories
 - **r** can see file names
 - **w** can add or remove files
 - **x** allow access to files in the directory
 - access to inodes

File mode

- Special:User:Group:Other
 - Special - SetUID,SetGID,Sticky
 - User - File owner access
 - Group - Group access
 - Other - Access by all others

Special Bits

- SetUID
 - Run executable with UID of owner
- SetGID
 - Executable: Run with GID of owner
 - Directory: New files inherit directory group
- Stricky bit
 - Legacy: Keep executable in swap on exit
 - Directory: Only root or owner can delete or rename files
 - Files: Usually ignored

r and x

- Binary executables only need x
 - Executed directly by the kernel
- Shell scripts need r and x
 - Interpreter needs to read the file
- Directories
 - r required to list files
 - x required to access files

File mode examples

-rw-----	1	mscott	mscott	/home/mscott/.ssh/id_rsa
-rw-r--r--	1	root	root	/etc/issue
-rwxr-xr-x	1	root	root	/usr/bin/zip
-rwsr-xr-x	1	root	root	/usr/bin/passwd
-rwxr-sr-x	1	root	tty	/usr/bin/wall
drwxrws---	1	root	sales	/home/sales
drwxrwxrwt	25	root	root	/tmp

Manipulating owner & permissions

- `ls -l` show permissions
- `stat` show detailed file permissions
- `chmod` sets permissions (-R for recursive)
- `chown` change user[:group] (-R for recursive)
- `chgrp` change group (-R for recursive)

chmod symbolic

- [ugoa] [+-=] [rwxst]
 - **u**ser, **g**roup, **o**ther, **a**ll
 - + add, - remove, = set
 - **r**ead, **w**rite, **e**xecute, **s**et^{*}**i**d, **s**ticky
- Examples
 - u+w add write permission for user
 - a-x remove execute permission for all
 - ug=rw,o=r set user & group to rw, others to r

chmod numeric

644	r <u>w</u> -r--r--	u=rw,g,o=r
755	r <u>wx</u> r-xr-x	u=rwx,g,o=rx
0755	r <u>wx</u> r-xr-x	u=rwx,g,o=rx
4755	rws <u>r</u> -xr-x	u=rwxs,g,o=rx
2775	rwx <u>rwsr</u> -x	u=rwx,g=rwxs,o=rx
1777	rwx <u>rwxrwt</u>	ug=rwx,o=rwxt

umask

- Sets file creation mode mask
 - In bash this is a shell builtin command
 - Set on login
- umask sets permissions to mask
 - If the bit is set in umask, it is 0 in the file mode
 - Result is perm & ~umask
- umask is often 0002 (o-w) or 0022 (go-w)

Group access

- Group access is critical to several labs
 - Allow users to share resources
 - Read access to files
 - Write access to files
- Groups define users with similar privileges
 - Shared access
- SetGID and umask determine defaults

Access Control Lists (ACL)

- Set permissions for individual users
 - `setfacl -m mscott:rwx /home/accounting`
- Show ACL
 - `getfacl /home/accounting`
 - A trailing + in `ls -l` indicates presence of ACL
- Not all filesystems support ACLs
 - ACL support can be set by mount
 - Should be a very last resort
- **If you resort to ACL in the labs, you are doing it wrong!!!**

setcap/getcap

- **capabilities** are privileges normally reserved for root that can be selectively granted to executables
 - `setcap cap_net_raw=ep command`
command can access raw sockets without being root
 - `getcap command`
show capabilities associated with command
- More selective than setuid

• Example capabilities

man capabilities show all capabilities

- CAP_CHOWN
 - Change file ownership
- CAP_KILL
 - Send any signal
- CAP_NET_RAW
 - Access to raw sockets
- CAP_SYS_RAWIO
 - Acces to hardware
- CAP_SYS_TIME
 - Access to real time clock
- CAP_SYSLOG
 - Access to logging

Root Access

Linux System Administration
Fall 2023

Being root

- There are no restrictions on what root can do
- When you are root, a typo can brick the system
- If root login is compromised, access is complete

Playing it safe

- Limit access to root login
 - Disable account or limit account to local logins
- Grant root access as needed with sudo
 - Fine grained control on commands
- Grant access via trusted commands
 - Special permission bits
 - Special file capabilities

Permissions Reserved for root

- Mounting a filesystem
- Changing a file or folder user & group ownership
- Bypassing the kernel to obtain or send network traffic on the physical wire itself
- Listening on TCP / UDP ports lower than 1024
- Controlling systemd
 - Reboot, start/stop system processes, etc.

Limiting root logins

- Set root login shell to /sbin/nologin
- PAM
 - /etc/pam.d/system-auth
 - /etc/pam.d/sshd
 - /etc/security/access.conf
- /etc/ssh/sshd_config
 - PermitRootLogin
 - no
 - yes
 - prohibit-password
 - forced-commands-only

root login policy

- No root access is the most restrictive
 - Could be a problem if the machine is hung
- No ssh logins is good for DMZ or untrusted net
 - Console access can be a useful backup
- Limited ssh logins
 - Good for remote administration

sudo



- su "do" (super user do)
- Preferred way of elevating permissions
 - Allow by user and by system command access
 - Logging of commands
- Periodically require password
- Partly developed at CU Boulder

sudo configuration

- Main configuration
 - /etc/sudoers
- Local configuration
 - /etc/sudoers.d/*
 - File name does not matter
 - Choose by function
 - Facilitates distributed management
- Edit configuration files using visudo

sudo environment

- env_reset
 - Limits variables from user shell transferred to root
- secure_path
 - Modifies PATH used to find commands
- Integrates with PAM

Permitted Commands

- Allow root and wheel (or sudo) to do anything
 - root ALL=(ALL) ALL
 - %wheel ALL=(ALL) ALL
- Generic format
 - user host = command
 - user host = (user) command
 - % refers to a group
 - , for a list
 - ! to exclude
 - ALL means no restriction

Aliases

- **Host_Alias**

```
Host_Alias FILESERVERS = www, nfs
```

- **User_Alias**

```
User_Alias MANAGERS = mscott, jhalpert
```

- **Cmnd_Alias**

```
Cmnd_Alias START /usr/bin/systemctl start,/usr/bin/systemctl restart
```

- **Aliases are context sensitive**

Wildcards

- Match command string
 - Could be the command, parameter or files
 - Symbol matching, e.g. [0–9]
 - Wildcards
 - * zero or more characters
 - ? zero or one characters
- Command allowed if it matches any line in sudoers
- Matches are BNF not regexes
 - Exclude matches with !

Wildcard examples

- Allow mscott access to /var/log/messages and backups
 - mscott ALL=/bin/cat /var/log/messages
 - mscott ALL=/bin/cat /var/log/messages.[1-9]
- Never end a command in *
 - Allows users to sneak additional stuff at the end of the command
 - mscott ALL=/bin/cat /var/log/messages*
allows
 - sudo cat /var/log/messages.1 and
 - sudo cat /var/log/messages /etc/shadow

No password

- Execute a command with elevated privileges without requiring the usual sudo password

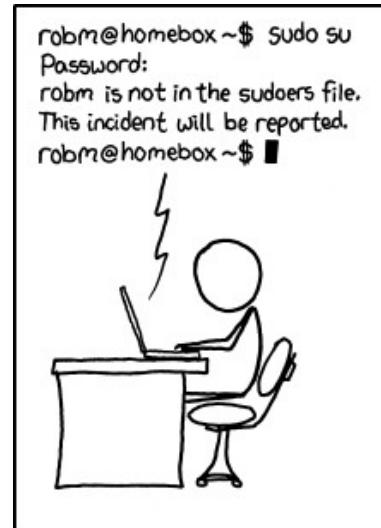
```
mscott ALL=NOPASSWD: /usr/bin/systemctl restart apache2
```

- ***Use with caution!!!***

- Password encourages the user to think before acting
 - Limit the command to the absolute minimum

sudo logging

- Commands run using sudo are logged
 - debian /var/log/auth.log
 - redhat /var/log/audit/audit.log
- Remote logging possible



sudo tips

- Test your configurations thoroughly
 - Could be a significant security hole
- Use aliases to simplify user experience
- Maintain customization in /etc/sudoers.d/
 - Works best with distributed management

Executables with privilege escalation

- setuid
 - run using UID of file owner of executable
- setgid
 - runs using GID of file group of executable
- setcap
 - Allow executable access to specific facilities otherwise requiring root

Extra Permission Bits

NUMBER	PERMISSION	RWX
7	setuid, setgid, stickybit	111
6	setuid and setgid	110
5	setuid, and stickybit	101
4	setuid	100
3	setgid, and stickybit	011
2	setgid	010
1	stickybit	001
0	none	000

Changing your password

- **/etc/shadow**
 - Ownership root : shadow
 - Permissions -rw-r-----
- **/usr/bin/passwd**
 - Ownership root : root
 - Permissions -rwsr-xr-x
 - Program is trusted to change the right user

Secure Shell

Linux System Administration
Fall 2023

Remote System Access

- The goal
 - Execute commands on a remote system
 - Remote shell or single command
 - Transfer files between systems
- Requirements
 - Invisible - terminal window just like local
 - Secure - the network cannot be trusted

In the beginning

- telnet (**teletype network**)
 - Original TCP/IP based remote terminal (RFC 15)
 - Plain text with lightweight escape sequences (0xff=IAC)
 - Port 23
 - Unsafe for untrusted networks
 - Still useful in testing TCP/IP connections
 - Sometimes enabled in legacy hardware
- Replaced by **secure shell (ssh)**

BSD r-commands

- Added in BSD 4.1 (1981)
 - Often remapped to ssh, scp, etc on many systems

Client	Function	Daemon	Port	Protocol
rexec	command	rexecd	512	TCP
rlogin	shell	rlogind	513	TCP
rsh	command	rshd	514	TCP
rcp	copy			
rstat	stat	rstatd		UDP
ruptime	uptime	rwhod	513	UDP
rwho	who			

Secure Shell (SSH)

- Designed by Tatu Ylönen 1995
 - OpenSSH - Implementation by OpenBSD (1999)
 - dropbear - Tiny footprint ssh for embedded devices
- Secure login to remote system
 - Public-key encryption over TCP port 22
 - Basis for scp, sftp, etc
 - Supports tunneling (X11, port forwarding)
 - Allows password-less logins

The importance of ssh

- Goal of exams are in large part establishing an ssh connection to a remote server
 - Basic machine functions are up
- Tunneling allow access to network services as if they are local

Using ssh

- ssh user@host
 - p port (default 22)
 - v verbose (-vv or -vvv for even more verbose)
 - L local:host:remote (tunnel)
 - R remote:host:local (reverse tunnel)
 - X Forward X11 (-Y for secure)

ssh examples

```
ssh www.dm.com
```

```
ssh -p 2222 root@broken.dm.com
```

```
ssh -v root@unreachable.dm.com
```

```
ssh -L 2222:10.21.32.2:22 machinea
```

```
ssh -p 2222 localhost
```

```
scp -Cp -P 2222 localhost:foo .
```

ssh files

- `/etc/ssh/ssh_config`
 - Global configuration
 - Can be by host or *
 - Override options with
`ssh -o`
- `/etc/ssh/sshd_config`
 - Incoming connections
- `~/.ssh/`
 - `known_hosts`
 - `signature`
 - `authorized_keys`
 - public key(s)
 - `id_rsa`, `id_ed25519`, ...
 - private key

Copying files with scp

- `scp localfiles host:remotedir`
`scp host:remotefiles localdir`
 - p preserve file times and modes
 - r recursive
 - P Port (default 22)
 - C Compress

Generating keys

- Generate key pair
 - ssh-keygen
 - t key type (rsa, rsa-sha2-512, dsa, ecdsa, ed25519, etc.)
 - b key length (e.g. 4096)
 - C comment
 - Default output to ~/.ssh/
 - Empty passphrase is OK
- ssh-keygen -t rsa -b 4096 -C saclass@colorado.edu
 - Private key ~/.ssh/id_rsa
 - Public key ~/.ssh/id_rsa.pub

Distributing keys

- By hand

```
scp ~/.ssh/id_rsa.pub remotehost:  
ssh remotehost  
cat id_rsa.pub >> ~/.ssh/authorized_keys  
rm id_rsa.pub
```

- `ssh-copy-id remotehost`

Using ssh in scripts

- ssh will only accept a password interactively
 - Requires password-less authentication
- sshpass allows ssh to be run non-interactively but with a password
 - PW=*secret*
`sshpass -p $PW ssh host command`
 - Could be a security hole

Notes about key pairs

- The local key must be in \$HOME/.ssh/
 - What is \$HOME when you do sudo?
- The remote key must be in \$HOME/.ssh/ for the remote user you are login in as

Pause for Paranoia

- Password-less logins extends the domain of trust to machines with the private key
 - Allows tunnels through firewalls
 - DMZ hosts should never have private keys for the LAN
 - Set up access from the LAN side
 - root password-less access should be minimized
 - Agonize over laptops

rsync

- Fast tool for copying files
 - Modern versions transparently run over ssh
 - Conditionally copies files based on time and size
 - Clever block checksum for updating files
 - Can preserve permissions, links and holey files
- Best tool to mirror directory trees

rsync options

- | | |
|----------------------------|------------------------------------|
| -a archive | -S write holey files |
| -p preserve permissions | -n dry run |
| -t preserve times | -v verbose |
| -r recursive | -z compress |
| -l preserve symbolic links | --delete-excluded |
| -o preserve owner | --existing (no new files) |
| -g preserve group | --ignore-existing (only new files) |
| -H preserve hard links | --exclude= <i>pattern</i> |
| -A preserve ACLs | --bwlimit= |

rsync tips

- Watch out for a trailing / on the source
- Use -n to do a dry run before big transfers
 - Especially with --delete-excluded
- *rsync* can be used locally as a clever *cp*

Copy trees with tar

```
ssh user@host tar cf - -C dir | tar xf -
```

- **ssh** to *host* and run **tar**

- C** does a chdir to *dir*

- cf** - creates and archive to stdout (-)

- **tar xf** -

- unpack archive from stdin (-)

- archive is piped over ssh

Important *sshd_config* options

- PermitRootLogin
 - yes = allowed
 - prohibit-password = must use key pair
 - forced-commands-only = key pair only, no shell
 - no = never
- UseDNS (should be yes)
- UsePAM (should be yes)

mosh

- mosh is an alternative to ssh that works better over unreliable connections
 - ssh takes a long time to recover from lost packets
 - mosh uses UDP
 - mosh tolerates changing the source IP address
- Good choice for mobile (cellular)
 - Example implementation: blink for iOS

Package Management

Linux System Administration
Fall 2023

In the beginning...

- Software on CD
- **autoconf** to deal with OS & compiler peculiarities
- Build from source

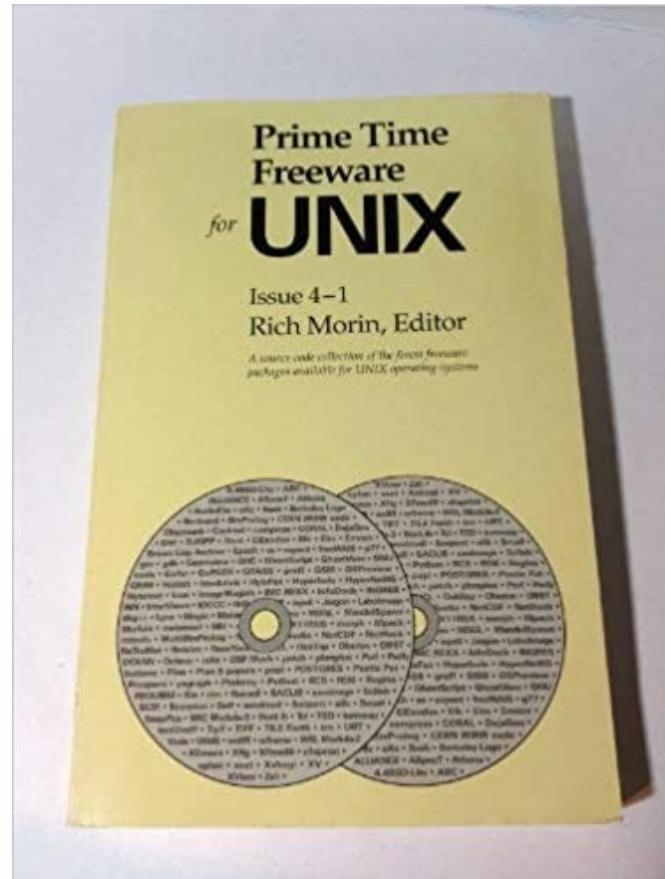
```
tar xzf apache_1.3.6.tar.gz
```

```
cd apache_1.3.6
```

```
./configure --prefix=/usr
```

```
make
```

```
sudo make install
```



Then came package managers

- Pre-compiled binaries and configuration files
- Checks shared library dependencies
- Specialized archive to include config scripts
 - Redhat rpm (.rpm)
 - Debian dpkg (.deb)
- CD or download



Package manager systems

- With the internet came repositories
 - Download on demand
- Package management systems download the package as well as its dependencies
 - Red Hat **yum** (now called **dnf**)
 - Debian **apt-get** (also **apt**)
 - Arch **pacman**
 - OSX **homebrew**
 - BSD **pkg**

Advantages

- Simplifies downloading and installing packages
- Facilitates management of dependencies
- Automate the process of updating packages
- Supports cleanly removing packages
 - but not necessarily dependencies

Package Repositories

- Collection of packages for a distribution
- Accessed via HTTP/HTTPS
- Mirrors spreads bandwidth load
 - Update mirrors with rsync
 - Many sites run a local mirror
- Repositories may have multiple distros and versions

RedHat EPEL

- Extra Packages for Enterprize Linux
 - Additional packages not included in RHEL
 - Adds more bleeding edge packages from Fedora
- **dnf install epel-release**
 - Add EPEL as an additional repository

Repository Configuration

- RedHat

/etc/yum.repos.d/*

- dnf still uses yum.repos.d

Rocky

/etc/yum.repos.d/rocky.repo

EPEL

/etc/yum.repos.d/epel.repo

- debian

/etc/apt/sources.list

deb *URL* release main

Package naming

- RedHat
 - package-version.os.arch.rpm
 - openssh-8.7p1-8.el9.x86_64.rpm
 - perl-base-2.27-479.el9.noarch.rpm
- debian
 - package_version+os_arch.deb
 - openssh-client_8.4p1-5+deb11u1_amd64.deb

Making sure all packages are current

- RedHat
 - dnf update
 - check metadata
 - check for updates
 - Confirmation defaults to NO
- debian
 - apt-get update
 - check metadata
 - apt-get upgrade
 - check for updates
 - Confirmation defaults to YES

Installing dig

- Redhat

dnf provides dig

- bind-utils

dnf install bind-utils

- Dependencies installed

bind-libs

bind-license

fstrm

libmaxminddb

libuv

protobuf-c

- debian

apt-file search /usr/bin/dig

- bind9-dnsutils

apt-get install bind9-dnsutils

- May already be installed

Installing a downloaded package

- rpm -i foo.rpm
- dpkg -i foo.deb
 - Could fail and show dependencies not met
- dnf install ./foo.rpm
- apt-get install ./foo.deb
 - Will fetch dependencies as needed

Finding out about packages

- Redhat
 - Find string in package name or description
 - `dnf search string`
 - Package metadata
 - `dnf info package`
- debian
 - Find string in package name or decription
 - `apt search string`
 - Package metadata
 - `apt show package`

What package installed dig?

- RedHat

```
rpm -qf /usr/bin/dig  
bind-utils-9.16.23-1.el9.x86_64
```

- debian

```
dpkg -S /usr/bin/dig  
bind9-dnsutils: /usr/bin/dig
```

What files were installed by bind-utils?

- RedHat

```
$ rpm -ql bind-utils  
...  
/usr/bin/arpaname  
/usr/bin/delv  
/usr/bin/dig  
/usr/bin/dnstap-read  
/usr/bin/host  
/usr/bin/nslookup  
/usr/bin/nsupdate  
...
```

- debian

```
$ dpkg-query -L bind9-dnsutils  
...  
/usr/bin/delv  
/usr/bin/dig  
/usr/bin/dnstap-read  
/usr/bin/mdig  
/usr/bin/nslookup  
/usr/bin/nsupdate  
...
```

What packages are installed

- RedHat
 - rpm -qa
- debian
 - dpkg -l
 - ii installed
 - un uninstalled

Removing packages

- Redhat
 - dnf remove *package*
 - rpm -e *package*
 - dnf autoremove
 - Removes obsolete
- debian
 - apt-get remove *package*
 - --purge removes configs
 - dpkg -r *package*
 - dpkg -P to purge
 - apt autoremove
 - Removes obsolete

Configuration files in packages

- Packages may contain configuration files
 - What happens if a new release changes the configuration file, but you changed the config?
- Redhat
 - Saves new file with .rpmsave or .rpmnew appended
- debian
 - Prompts to keep, replace, modify, etc.
- Generally you will need to re-apply your edits

Some sage advice

- When changing a configuration file, save a copy of the original
 - cp -a foo.conf foo.conf.0
 - diff shows changes
 - Helps to recover from failures
 - Reboot if appropriate
- RedHat does not enable packages automatically

Installing groups of packages

- Redhat
 - dnf groupinstall “Development Tools”
- debian
 - apt install build-essential
- Installs package collections
 - gcc, g++, make, binutils, header files, and lots more

Automated installation

- Bare metal installation from a script
 - Used in data centers or compute clusters
- RedHat/Anaconda kickstart
- debian FAI (Fully Automatic Installation)

Apache

Linux System Administration
Fall 2023

What is Apache?

- One of the original web servers
 - Initially released in 1998
- Stable, very configurable and widely used
- Part of LAMP
 - Linux
 - Apache
 - Mysql
 - PHP/Perl/Python

Alternatives to Apache

- nginx
 - more geared to serving static content
- lighttpd
 - good security and performance
- many others

Quick Start

- Redhat
 - dnf install httpd
 - Enable and start httpd
 - systemctl enable httpd
 - systemctl start httpd
 - Disable and stop firewall
 - systemctl disable firewalld
 - systemctl stop firewalld
 - or punch a hole in the firewall
 - firewall-cmd --zone=public --add-port=80/tcp --permanent
 - firewall-cmd --reload
- Debian
 - apt install apache2
 - Enabled and started by default
 - No firewall
- On your Virtualbox
 - Forward (say) 800x to port 80

Connect with a browser

- RedHat

HTTP Server Test Page

This page is used to test the proper operation of an HTTP server after it has been installed on a Rocky Linux system. If you can read this page, it means that the software is working correctly.

Just visiting?

This website you are visiting is either experiencing problems or could be going through maintenance.

If you would like to let the administrators of this website know that you've seen this page instead of the page you've expected, you should send them an email. In general, mail sent to the name "webmaster" and directed to the website's domain should reach the appropriate person.

The most common email address to send to is:
"webmaster@example.com"

Note:

The Rocky Linux distribution is a stable and reproducible platform based on the sources of Red Hat Enterprise Linux (RHEL). With this in mind, please understand that:

POWERED BY  POWERED BY 

- debian

Apache2 Debian Default Page

 It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Debian systems. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

Configuration Overview

Debian's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Debian tools. The configuration system is **fully documented in `/usr/share/doc/apache2/README.Debian.gz`**. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the `apache2-doc` package was installed on this server.

The configuration layout for an Apache2 web server installation on Debian systems is as follows:

```
/etc/apache2/
|-- apache2.conf
|   '-- ports.conf
-- mods-enabled
|   '-- *.load
|       '-- *.conf
-- conf-enabled
|   '-- *.conf
-- sites-enabled
    '-- *.conf
```

- `apache2.conf` is the main configuration file. It puts the pieces together by including all remaining configuration files when starting up the web server.
- `ports.conf` is always included from the main configuration file. It is used to determine the listening ports for incoming connections, and this file can be customized anytime.

HTTP or HTTPS?

- HTTP
 - Insecure
 - Port 80
 - Plain TCP/IP connection
- HTTPS
 - Secure
 - Port 443
 - TLS (was SSL)
 - Transport Layer Security
 - Requires PKI certs

Where do I find things?

- Redhat
 - Configuration
 - /etc/httpd
 - HTML files
 - /var/www/html
 - CGI files
 - /var/www/cgi-bin
 - Logs
 - /var/log/httpd
- debian
 - Configuration
 - /etc/apache2
 - HTML files
 - /var/www/html
 - CGI files
 - /usr/lib/cgi-bin
 - Logs
 - /var/log/apache2

Apache Security

- Web services are a common target
- The server runs as a non-priviliged user
 - Redhat apache
 - Debian www-data
- Directories must be readable by user or group
 - CGI programs sometimes need write access

Apache Concepts

- Static files are in the DocumentRoot
 - File are served verbatim
- Dynamic content is the result of running a program (CGI)
 - Common Gateway Interface
 - Request on `stdin`
 - Response on `stdout`
 - `stderr` to error log
- Modules to perform varous actions
 - Access control
 - Rewriting URLs
 - Proxy and load balancing
- Named and IP based Virtual web servers
 - One server, many sites
 - Highly parallel

Basic configuration

- Configuration files looks like HTML
 - <Directory "/var/www/cgi-bin">
 - <VirtualHost "foo.bar.com:80">
- RedHat puts everything in httpd.conf
 - Does include a files in /etc/httpd/conf.d
- debian spreads configuration over multiple files and directories
 - Provides a2en* a2dis* commands to modify
 - apachectl
 - -S shows content
 - -t checks syntax

Important Keywords

- ServerName FQDN for this host
- ServerRoot Root directory root for configuration
- Listen IP and port to bind to
- DocumentRoot Root directory for files
- SSLEngine Enable SSL/TLS
- SSL*Cert* Certificates

Directory Directives

- <Directory *path*> select directory
- Options **control** apache behavior for this path
 - Indexes **allow** directory listing
 - FollowSymLinks **follow** symbolic links
 - ExecCGI **allow** CGI
- AllowOverride **controls** .htaccess
- Require **set** authorization restrictions

HTTP Exchange

Request sent to TCP port 80

GET / HTTP/1.1

Host: dundermifflin.com

User-Agent: Mozilla/5.0

Accept: text/html;

Connection: keep-alive

Response

HTTP/1.1 200 OK

Date: Mon, 23 May 2005 22:38:34 GMT

Content-Type: text/html; charset=UTF-8

Content-Length: 155

Last-Modified: Wed, 08 Jan 2003 23:11:55 GMT

Server: Apache/1.3.3.7 (Red-Hat/Linux)

Connection: close

<html>.....</html>

Adding a virtual Host

```
<VirtualHost *:80> IP adddress (*) and port  
  ServerName    FQDN of virtual host  
  ServerAlias   Alternative FQDN of virtual host  
  ServerAdmin   email of admin  
  DocumentRoot  directory containing HTML files  
  ErrorLog      error log file  
  CustomLog     access log file  
</VirtualHost>
```

Rewriting Rules

- Permanent remap (HTTP to HTTPS)
 - Redirect permanent / https://dm.com/
- Replace URL
 - RewriteEngine on
 - RewriteRule ^/?old([a-zA-Z.]*)\$ /new\$1 [R=301,L]
 - Changes oldxxxxx to newxxxxx
- Requires mod_rewrite

icons

- Images are stored in /usr/share
 - Debian /usr/share/apache2/icons
 - RedHat /usr/share/httpd/icons
- Used in index and HTML files
- Varies between machines

CGI scripts

Python

```
#!/usr/bin/python

import os;
print("Content-type: text/html\r\n\r\n");
print("<H2>Environment</H2>\r\n");

for param in os.environ.keys():
    print("%s=%s<br>\n" % (param,os.environ[param]));
```

Perl

```
#!/usr/bin/perl

print "Content-type: text/html\r\n\r\n";
print "<H2>Environment</H2>\r\n";

while (my ($key,$val) = each %ENV)
{
    print "$key=$val<br>\n";
```

Hardening Apache

- fail2ban
 - blocks IP using iptables based on logs
- selinux (RedHat)
- apparmor (Debian)
 - limits Apache access to certain directories

Scheduling Tasks

Linux System Administration
Fall 2023

Scheduling Commands

- cron
 - Run a recurring command
- at
 - Run a command at a specified time
- systemd timers
 - Re-invents the wheel

Common periodic tasks

- Rotate logs
- Check for software updates
- Perform backups

cron files

- One liners
 - /etc/crontab system-wide entries
 - /etc/cron.d additional crontab entries
- Scripts
 - /etc/cron.hourly script that run hourly
 - /etc/cron.daily scripts that run daily
 - /etc/cron.weekly scripts that run weekly

crontab entries

min hour dom mon weekday user command

- * = don't care
- */int = match every (*/10 = 0, 10, 20, 30, 40, 50)
- integer = match exactly
- int-int = range (1-5 = 1, 2, 3, 4, 5)
- int,int,int = match from list

Sunday is weekday 0

crontab examples

- At 00:15 each day, rsync backup from machinea

```
15 0 * * * root rsync -aH machinea:/backup/* /backup
```

- Run the checkwx script on the hour

```
0 * * * * root /usr/local/bin/checkwx
```

- Run the ntsgw script every 5 minutes

```
*/5 * * * * willem /usr/localbin/ntsgw
```

- Run the backup-db script Sunday morning at 2am

```
0 2 * * 0 root /usr/local/bin/backup-db
```

- Run the snapshot script at 00:30 on the first of the month

```
30 0 1 * * root /usr/local/bin/snapshot
```

at

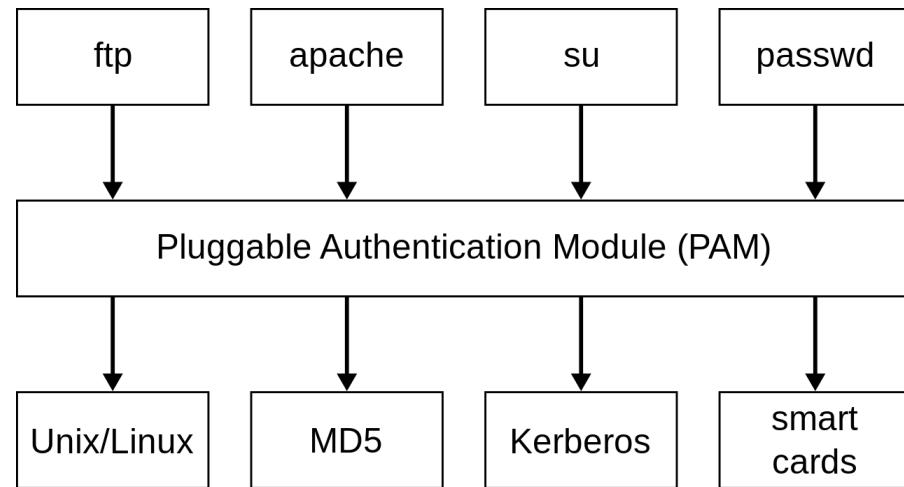
- Commands to run are read from the command line and executed as /bin/sh
 - at 21:00 at 9pm
 - at now+20 minutes (20 minutes from now)
- atq show jobs queued
- atrm remove job from queueq

Pluggable Authentication Modules (PAM)

Linux System Administration
Fall 2023

What is PAM?

- Common authentication module used by many services to authenticate users
 - Allows fine grained control for user access
 - Single service to simplify authentication



Important Password Attributes

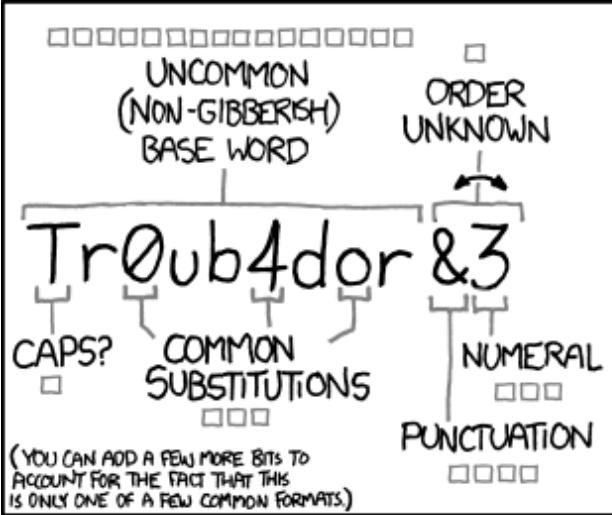
- Username
- Password hash (Method + Salt + Hash)
- Last date changed
- Maximum age
 - Warning time
 - Inactive time
 - Expire

Password Threats

- Hashes from /etc/shadow or elsewhere
- Plain-text passwords in scripts
- Sniffing of insecure protocols
- Shoulder surfing
- Keyboard loggers (hardware or software)
- Brute force guessing or dictionary attacks

Mitigating password threats

- Exclude nonessential users from the machine
- Enforce passwords that are randomly generated, 12 characters or longer and from a set of 94 characters, symbols and numbers
- Using a key fob or application to generate key
- Two factor authentication
- Biometrics



~28 BITS OF ENTROPY

$2^{28} = 3$ DAYS AT 1000 GUESSES/SEC

(PLAUSIBLE ATTACK ON A WEAK REMOTE WEB SERVICE. YES, CRACKING A STOLEN HASH IS FASTER, BUT IT'S NOT WHAT THE AVERAGE USER SHOULD WORRY ABOUT.)

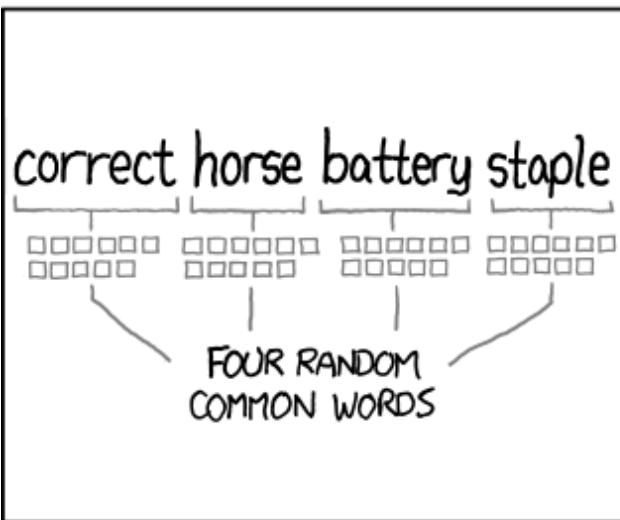
DIFFICULTY TO GUESS: EASY

Detailed description: This panel shows a password with approximately 28 bits of entropy. It includes a calculation ($2^{28} = 3$ DAYS AT 1000 GUESSES/SEC) and a note about a plausible attack on a weak remote web service.

WAS IT TROMBONE? NO, TROUBADOR. AND ONE OF THE 0s WAS A ZERO?
AND THERE WAS SOME SYMBOL...

Detailed description: This panel features a stick figure thinking. The thought bubble contains the text 'WAS IT TROMBONE? NO, TROUBADOR. AND ONE OF THE 0s WAS A ZERO?' and 'AND THERE WAS SOME SYMBOL...'. The stick figure has a small circle above its head.

DIFFICULTY TO REMEMBER: HARD



~44 BITS OF ENTROPY

$2^{44} = 550$ YEARS AT 1000 GUESSES/SEC

DIFFICULTY TO GUESS: HARD

Detailed description: This panel shows a password with approximately 44 bits of entropy. It includes a calculation ($2^{44} = 550$ YEARS AT 1000 GUESSES/SEC) and a note that it is 'HARD' to guess.

THAT'S A BATTERY STAPLE. CORRECT!

Detailed description: This panel features a stick figure thinking. The thought bubble contains the text 'THAT'S A BATTERY STAPLE.' and 'CORRECT!'. Below the thought bubble is a drawing of a battery with a plus sign (+) and a minus sign (-). The stick figure has a small circle above its head.

DIFFICULTY TO REMEMBER: YOU'VE ALREADY MEMORIZED IT

THROUGH 20 YEARS OF EFFORT, WE'VE SUCCESSFULLY TRAINED EVERYONE TO USE PASSWORDS THAT ARE HARD FOR HUMANS TO REMEMBER, BUT EASY FOR COMPUTERS TO GUESS.

Common sense password policy

- Enforce passwords that are strong enough, and educate them on selecting a good memorable password and force them to change it periodically.
- Use two factor authentication for critical systems

Password Expiration Example

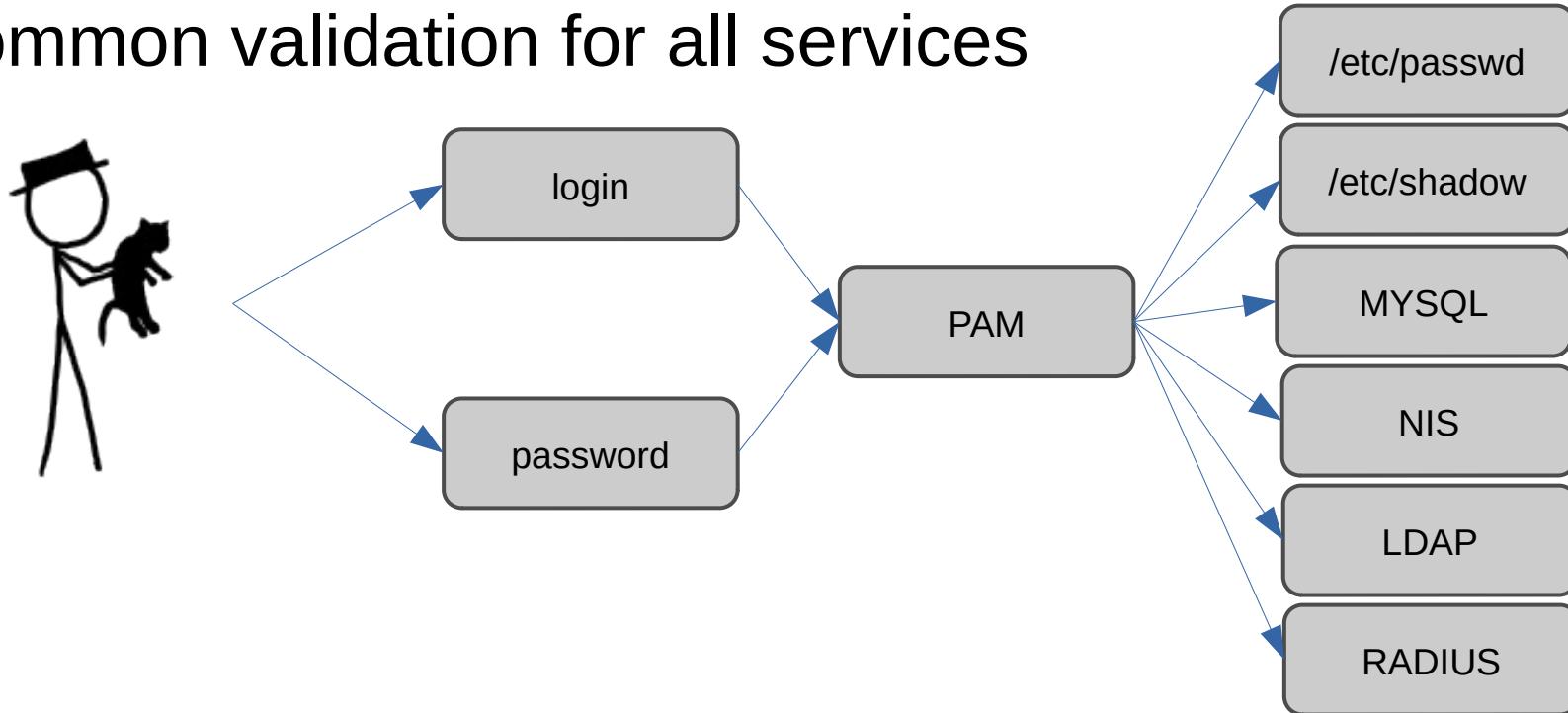
- mscott:\$6\$88afba...:19245:0:90:10:7::
 - 19245 = Sep 10, 2022
 - 90 = Maximum age (Dec 9, 2022)
 - 10 = Warning period (Nov 29, 2022)
 - 7 = Inactivity period (Dec 15, 2022)
 - account expiration period (never)

Setting password aging

- `chage user`
 - l show current settings
 - d 0 force change at next login
 - E set expiration date
 - l lock account
 - u unlock account

PAM login

- Pluggable Authentication Modules
- Common validation for all services



PAM Modules

- Authorization
 - Identifies users, groups and membership
- Account
 - Enforces login restrictions
- Session
 - Takes cares of tasks like creating or mounting home directory
- Password
 - Deals with password changes

PAM Files

- /etc/pam.d
 - PAM configuration files
- /etc/security
 - Policy files
 - access.conf - who can login from where
 - pwquality.conf - password requirements

Enable PAM Access

- RedHat
 - Files
 - /etc/pam.d/system-auth
 - /etc/pam.d/password-auth
 - Add after last account entry
 - account required pam_access.so
- Debian
 - Files
 - /etc/pam.d/login
 - /etc/pam.d/sshd
 - Uncomment
 - account required pam_access.so

This enables /etc/security/access.conf

/etc/security/access.conf

- Format
 - Allow/Deny : user or group : from
- Evaluated in order
 - + :root :ALL allow root from anywhere
 - + :mscott :10.21.32.2 allow mscott from machinee
 - + : (sales) :ALL allow group sales from anywhere
 - :ALL :ALL deny all others

PAM Password Quality Checks

- Is it a dictionary word?
- Is it a palindrome?
- Did only the case change?
- Is it too similar to the old one?
- Is it just a rotated version of the old one?
- Is it too simple?
- Are there more than 5 different characters?
- Does it contain the username or gecos?

/etc/security/pwquality.conf

- minlen **minimum length**
 - difok **difference from old**
 - gecoscheck **no gecos**
 - dictcheck **no words**
 - usercheck **no username**
 - maxrepeat **repeat limit**
 - minclass **min # classes**
- **Classes**
 - Positive is a max credit
 - Negative is a minimum
 - dcredit **digits**
 - ucredit **upper case**
 - lcredit **lower case**
 - ocredit **other**

Password quality examples 1

- Required
 - dcredit = 2
 - lcredit = 0
 - minlen = 10
- Proposed
 - r2wrexac too short (8+1)
 - r2wrexal OK (8+2)
 - r2wre32 too short (7+2)
- Required
 - dcredit = -2
 - lcredit = 0
 - minlen = 10
- Proposed
 - r2wrexacpw only one digit
 - r2wrexal too short
 - r2wrexal32r OK

Password quality examples 2

- Required
 - dcredit = 2
 - ucredit = 1
 - lcredit = 0
 - minlen = 10
- Proposed
 - r2wrexac OK (8+1+1)
 - r2wrexal OK (8+2)
 - r2wRe32 OK (7+2+1)
- Required
 - dcredit = -2
 - lcredit = 0
 - minlen = 10
 - minclass = 3
- Proposed
 - r2wrexacpw only one digit
 - r2wrexal32r only 2 classes
 - R2wrexal32r OK

Configuration Management

Linux System Administration
Fall 2023

How to manage multiple machines

- Scripts that configure users and services
 - Very flexible, can accommodate heterogeneity
 - Ad hoc procedures can be confusing
- Configuration Management tools
 - Declarative desired state of each machine
 - Knows common tasks like user & network configuration
 - Manage large number of machines

CM goals

- idempotent
 - Applying multiple times does not create duplicates
- cross platform
 - Works across flavors of Linux or Unix
 - Some also support Windows
- distributed
 - Apply to all managed machines

Commonly used CM systems

System	Initial Release	Implementation	Configuration	Web site
CFEngine	1993	C	custom	cfengine.com
Puppet	2005	Ruby	custom	puppet.com
Chef	2009	Ruby	Ruby	chef.io
Salt	2011	Python	YAML	saltstack.com
Ansible	2012	Python	YAML	ansible.com

and many more...

https://en.wikipedia.org/wiki/Comparison_of_open-source_configuration_management_software

Using a CM

- Pros
 - Consistent configuration on all hosts
 - Declarative syntax
 - Cross platform
 - Remote management
 - Easier to document
- Cons
 - Errors can take down many hosts
 - CM differ in approach and terminology
 - Manual config/CM conflicts makes ad hoc changes difficult

CM User Example

- Puppet

```
user { "mscott":  
    uid => "2001",  
    ensure => present,  
    comment => "Michael Scott",  
    gid => "mscott",  
    groups => ["managers"],  
    shell => "/bin/bash",  
    home => "/home/mscott",  
    managehome => true,  
}
```

- Ansible

```
name: User Michael Scott  
ansible.builtin.user:  
    name: mscott  
    uid: 2001  
    state: present  
    comment: "Michael Scott"  
    group: mscott  
    groups: managers  
    shell: /bin/bash  
    home: /home/mscott  
    create_home: yes
```

Installing Puppet

- Install package on all machines
 - dnf install puppet
 - apt install puppet
- Install modules
 - puppet module install puppet-network
- Apply a puppet manifest
 - puppet apply dmusers.pp

Puppet Modules

- Public repo <https://forge.puppet.com>
 - Quality and age varies by module
- Builtin modules for basic things
 - files and directories
 - users & groups
- Puppet can be run as client/server
 - Requires setting up puppetserver and certs
 - Manifests often just copied and then applied

Ansible Concepts

- Control node used to manage network
- Managed nodes
 - Accessed via ssh
 - Does not have ansible installed
- Module (code to accomplish tasks)
- Playbook (things to accomplish)
 - YAML file
 - Task (what we want to accomplish)

Installing Ansible

- Ansible need only be installed on the control node

```
dnf -y install ansible or apt -y install ansible
```

- Access to managed hosts is by ssh

- Set yourself up for password-less logins

- Configure ansible

- /etc/ansible/ansible.cfg **add** pipelining=true

- /etc/ansible/hosts **configure** saclass hosts

Applying ansible

- Show hosts in saclass group
 - ansible saclass --list-hosts
- Check connectivity
 - ansible saclass -m ping
- Running privileged commands
 - ansible-playbook -K umask.yaml
 - -K prompt for the BECOME password

Play settings

name: Name of the play

hosts: Hosts to apply this play to

remote_user: The ssh user on the managed node

become: Run command on managed node as another

become_user: User to run remote command as

tasks: What to do

Running a play

```
vlakkies@machinee $ ansible-playbook -K umask.yaml
BECOME password:

PLAY [default umask]
*****
TASK [Gathering Facts]
*****
ok: [100.64.41.6]
ok: [100.64.41.3]
ok: [100.64.41.2]
ok: [100.64.41.4]
ok: [100.64.41.1]

TASK [Set /etc/profile.d/umask.sh] ****
changed: [100.64.41.3]
changed: [100.64.41.6]
changed: [100.64.41.1]
changed: [100.64.41.2]
changed: [100.64.41.4]

PLAY RECAP ****
100.64.41.1      : ok=2    changed=1    unreachable=0    failed=0     skipped=0    rescued=0    ignored=0
100.64.41.2      : ok=2    changed=1    unreachable=0    failed=0     skipped=0    rescued=0    ignored=0
100.64.41.3      : ok=2    changed=1    unreachable=0    failed=0     skipped=0    rescued=0    ignored=0
100.64.41.4      : ok=2    changed=1    unreachable=0    failed=0     skipped=0    rescued=0    ignored=0
100.64.41.6      : ok=2    changed=1    unreachable=0    failed=0     skipped=0    rescued=0    ignored=0
```

Security considerations

- Relies on ssh to access remote machines
 - Requires one credential for access
 - Requires one credential for sudo access
 - An action should require ONE password
- Small site: Use sysadmin credentials
- Large site: Some reasonable compromise

Ansible builtin modules

- `ping` verify basic connectivity
- `user` manage user accounts
- `group` manage user groups
- `file` manage files and properties
- `copy` copy files
- `template` copy a file based on a template
- `systemd` manage systemd services
- `cron` manage crontab and cron.d entries
- `dnf` manage packages using dnf
- `apt` manage packages using apt

Booting and Services

Linux System Administration
Fall 2023

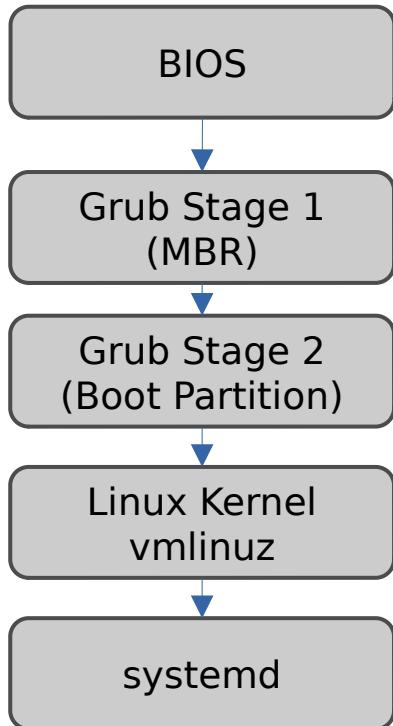
Bootstrapping

- Loading initial software onto the hardware

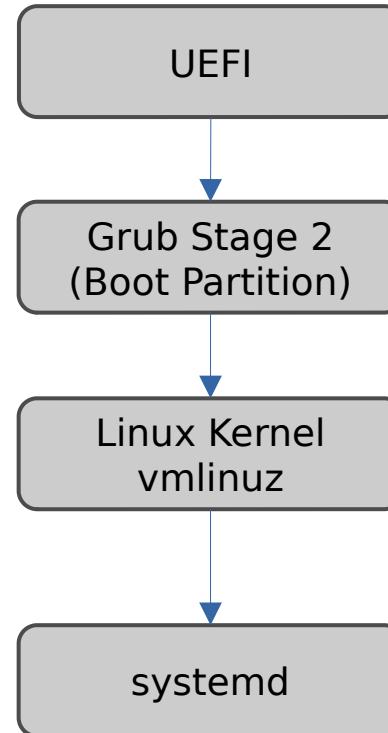


Modern Bootstrap

- Legacy BIOS



- UEFI Boot



Boot Methods

- BIOS
 - **Basic Input/Output System**
- UEFI
 - **Unified Extensible Firmware Interface**
- PXE
 - **Preboot Execution Environment**
 - Part of UEFI

Bootloaders

- loadlin
 - Original linux bootloader (obsolete)
- lilo and elilo
 - **l**inux **l**oader (legacy)
- GRUB
 - **G**Rand **U**nified **B**ootloader
- syslinux
 - group of loaders for CD, USB, etc.

GRUB

- BIOS started from ROM
 - Stage 1 loaded from Master Boot Record
 - Stage 2 loaded from /boot
- UEFI started from ROM
 - Stage 2 loaded from /boot
- Reads /boot/grub/grub.cfg (made from /etc/grub.d)
 - Which kernel to load
 - Parameters to pass to the kernel
 - Runs vmlinuz

Loading the kernel

- vmlinuz compressed Linux kernel
- initramfs or initrd.img
 - Initial RAM disk file system for kernel
- config kernel configuration stuff
- System.map kernel crash symbol maps

Kernel parameters

- Check running kernel `cat /proc/cmdline`
- Edit by `e` on GRUB boot prompt
 - `root=` root file system (`UUID=` or `/dev/????`)
 - `ro` mount read only
 - `quiet` quiet boot
 - `systemd.unit=` set systemd target
 - `rd.break` start shell instead of systemd
 - `init=` process to launch

Managing processes

- init
 - Traditional UNIX mother of all processes
 - PID=1
 - Used by BSD and legacy and rebel Linux distros
 - Runlevels 0-6
 - rc scripts
- systemd
 - New parallel process manager
 - PID=1
 - RedHat, Debian, ...
 - targets
 - units
 - too many other things

init runlevels

- Runlevel use
 - 0 system halt
 - 1 single user
 - 2 multi-user
 - 3 multi-user with networking
 - 4 no standard
 - 5 graphical login
 - 6 reboot
- Set as init N
- Legacy Linux/System V
 - Scripts in /etc/init.d
 - start, stop, restart, reload
 - Linked to /etc/rcN.d
 - /etc/rc2.d/K80apache2
 - Kills apache2 at runlevel 2
 - /etc/rc3.d/S20apache2
 - Start apache2 at runlevel 3
 - Sort determines order
- BSD somewhat different

systemd concepts

- target
 - More flexible than runlevels
 - Names have meaning
 - rescue
 - multi-user
 - graphical
 - Dependency tree
 - Can be user defined
- unit
 - Defines a service, device,...
 - Explicit dependencies
 - Wants start in parallel
 - Requires start after
 - Also Before and After
 - WantedBy when to start

systemd files

- `/lib/systemd` **systemd code**
- `/lib/systemd/system` **unit and target files**
- `/etc/systemd` **configuration files**
- `/etc/systemd/system/* .wants` **enabled units**
 - **symbolic links to** `/lib/systemd/system`
- `/bin/systemctl` **control program**

Example unit: apache2.service

```
[Unit]
Description=The Apache HTTP Server
After=network.target remote-fs.target nss-lookup.target
Documentation=https://httpd.apache.org/docs/2.4/

[Service]
Type=forking
Environment=APACHE_STARTED_BY_SYSTEMD=true
ExecStart=/usr/sbin/apachectl start
ExecStop=/usr/sbin/apachectl graceful-stop
ExecReload=/usr/sbin/apachectl graceful
KillMode=mixed
PrivateTmp=true
Restart=on-abort

[Install]
WantedBy=multi-user.target
```

Example Unit: ssh.service

```
[Unit]
Description=OpenBSD Secure Shell server
Documentation=man:sshd(8) man:sshd_config(5)
After=network.target auditd.service
ConditionPathExists=!/etc/ssh/sshd_not_to_be_run

[Service]
EnvironmentFile=/etc/default/ssh
ExecStartPre=/usr/sbin/sshd -t
ExecStart=/usr/sbin/sshd -D $SSHD_OPTS
ExecReload=/usr/sbin/sshd -t
ExecReload=/bin/kill -HUP $MAINPID
KillMode=process
Restart=on-failure
RestartPreventExitStatus=255
Type=notify
RuntimeDirectory=sshd
RuntimeDirectoryMode=0755

[Install]
WantedBy=multi-user.target
Alias=sshd.service
```

Managing systemd

- `systemctl` **action unit**
 - `start` **start immediately**
 - `stop` **stop immediately**
 - `restart` **stop&start immediately**
 - `enable` **set to start on boot**
 - `disable` **do not start on boot**
 - `status` **show status**
 - `is-enabled` **will it start on boot?**
 - `is-active` **is it running?**
 - `daemon-reload` **restart systemd**
 - `isolate` **activate target**
 - `show units`
 - `list-dependencies` **show tree**
 - `mask` **prohibit unit from starting**
 - `default = isolate` **default**
 - `set-default` **set default target**
 - `get-default` **show default target**

Troubleshooting systemd

- Check if a service is running and enabled
- Stop, start, restart, enable, disable service
- Recognize, get and set the default target
- Isolate a target

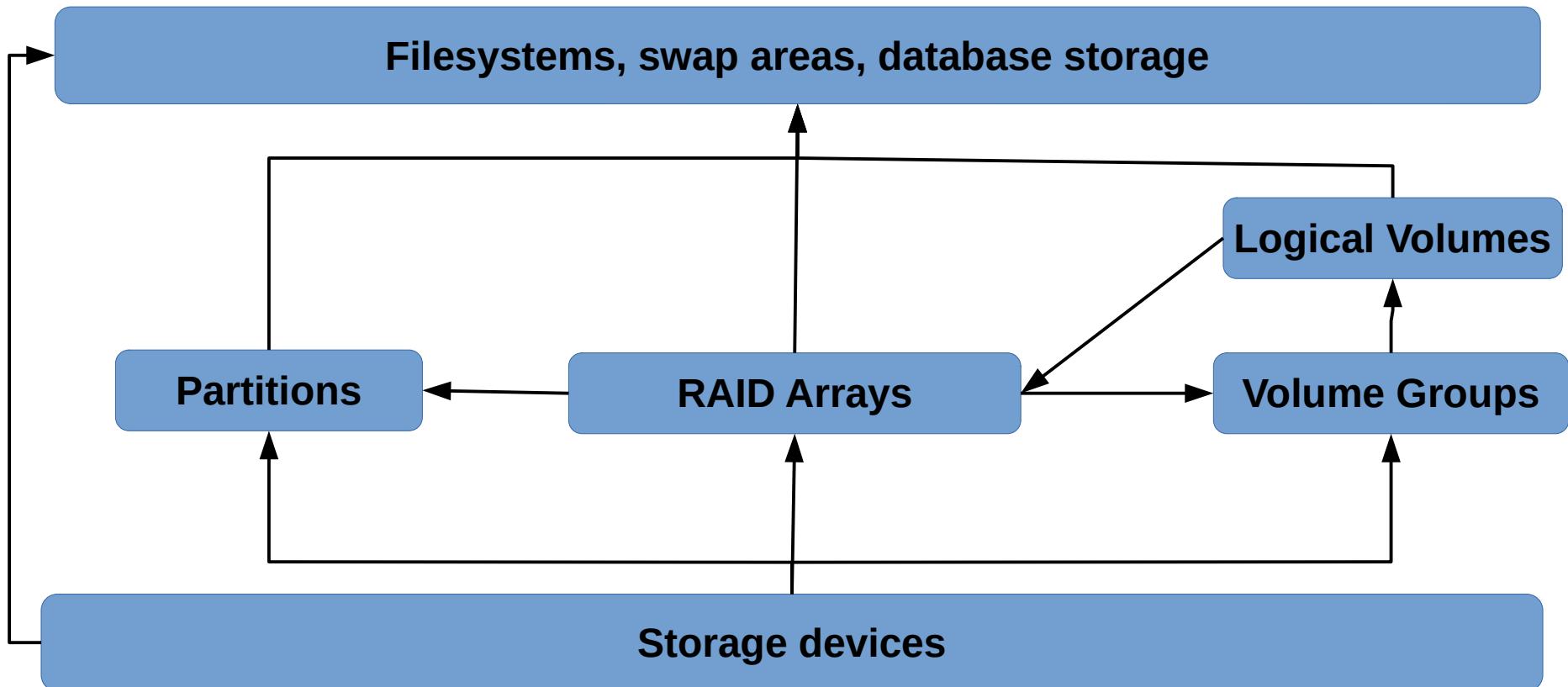
Filesystems

Linux System Administration
Fall 2023

Filesystems

- On Unix/Linux, everything is a file
- Files are organized in a directory tree
- A file system is what stores files
 - File systems integrate into the directory tree
- A disk is a physical storage device
 - Storage blocks are called sectors
 - Disk can be divided into partitions
 - Disks can be grouped in RAID arrays

Storage Management

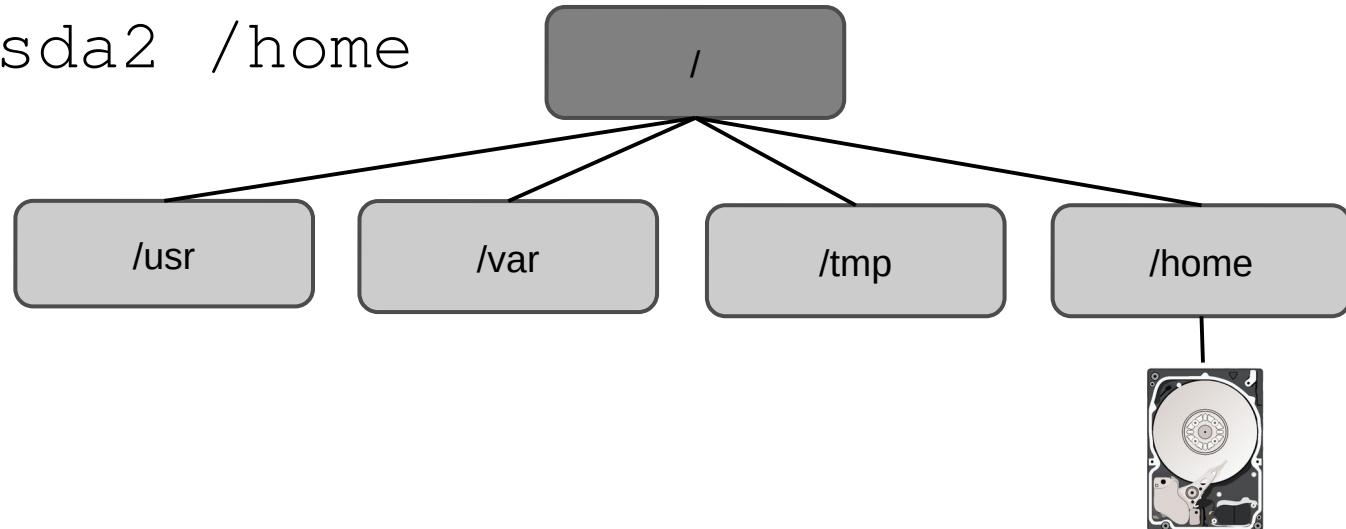


Useful filesystem separation

- /boot separate partition for grub
 - /boot/efi separate partition for grub/efi
- /tmp or /scratch temporary storage
- /home user directories
- /var contains cache, mail, run, etc
- /var/log contains system logs
- /var/lib/mysql database storage

Mounting filesystems

- mount device dir
 - The mount point must be an existing directory
 - The file system becomes part of the tree
 - Existing sub-tree becomes inaccessible
 - mount /dev/sda2 /home



Unix file system concepts

- Derived from the original Fast File System (FFS)
 - Disks are block devices
 - Directory tree structure
 - Directory entries point to inodes
 - inodes contain file characteristics
 - Type, permissions, times, etc.
 - Pointers to data blocks & indirection
 - Superblocks, cylinder groups, fragments
- Journaling added much later

File system features to consider

- Stability & robustness
 - Detecting bit rot
 - Safe handling of power or system failure
- Ability to store very large and very small files
- Performance
 - Read and write speed
 - Recovery time after failure
- Snapshots, replication, compression, encryption, etc.

Popular file systems

- UFS Unix File system (FFS)
- ext2/ext3/ext4 Extended File System
 - ext3 added journaling (backwards compatible)
- XFS extent based file system
- ZFS incorporates RAID and volume manager
- Btrfs B-tree file system("ZFS lite")
- tmpfs volatile memory file system
- *FAT* File Allocation Table (1977 floppy disks - current USB)

File systems can be mixed

- vfat for /boot/efi to simplify booting
- ext4 for / good for general purpose use
- xfs for /home to store large media files
- BrtFS for /var/lib/mysql for snapshots
- NFS4 for /var/mail network shared storage
- File system type and mount point are seamless
 - df -T or cat /etc/fstab shows the detail

Device Naming

- `/dev/hdX` Legacy hard drives
- `/dev/sdX` SCSI (SATA, SAS, USB) drives
- `/dev/srX` CD/DVD drives
- X is in the order of discovery a, b, c, d, ..
- Partitions are appended numerically
 - `/dev/sdc1` First partition (1) on third drive (c)

Persistent file names

- **UUID/GUID Universally/Globally Unique ID**
 - Stored on disk in file system superblock
 - `/dev/disk/by-uuid/*` symbolic link to `/dev/*`
- `blkid` shows UUID and other information
- UUID can be referenced in `/etc/fstab`

/etc/fstab

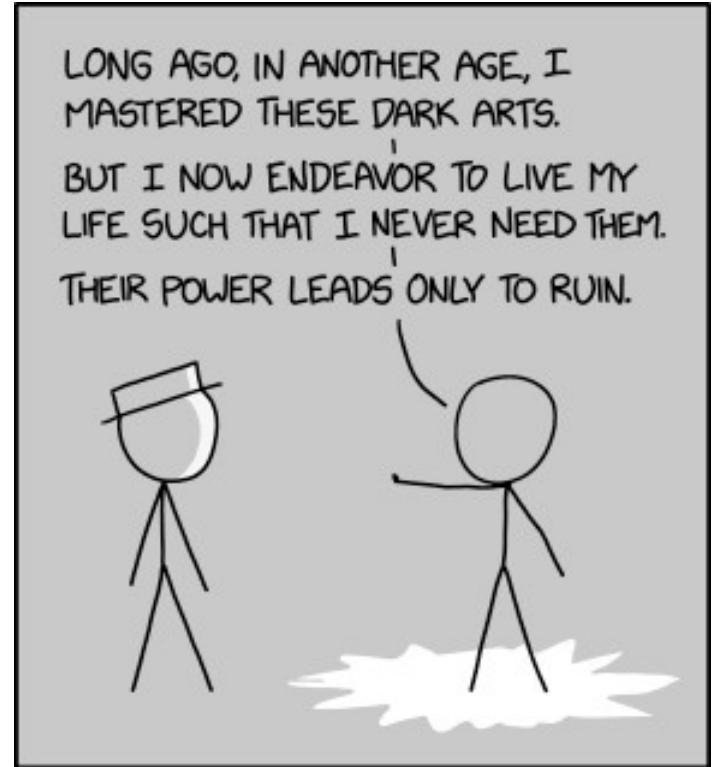
- Disks to mount on boot (or `mount -a`)
 - **device** `mount type options backup check`
 - `/dev/sda1 / ext4 errors=remount-ro 0 1`
 - **device** name of the device
 - `/dev/sda1 UUID=4c3f0e41-9bec-449c-8c57-95b5a4cb9927 /dev/mapper/root`
 - **mount point** where this go in the directory tree
 - `/` or `/home` or `/var/lib/mysql`
 - **type** file system type
 - `ext4` or `xfs` or `tmpfs` or `nfs4` or `swap`
 - **options** mount options
 - `errors=` or `ro` or `rw` or `sw` and may others
 - **backup** (obsolete - used by `dump` - set to zero)
 - **check** file system check order
 - `0=no fsck, 1=fsck first (file root), 2=fsck next`

Filesystem commands

- `mount` mount a filesystem in the tree
- `umount` unmount a filesystem
- `fsck` file system check
 - Generic front end for `fsck.ext3`, `fsck.xfs`, etc
- `mkfs` create a new file system (**D.F.I.U.**)
 - Generic front end for `mkfs.ext3`, `mkfs.xfs`, etc
- `badblocks` check drive for bad blocks (low level)

Scripting should be done with caution

- If something goes wrong, the remainder of the script can do a LOT of damage
- If you forget to update some parameters, you can destroy the system



MY RESPONSE WHENEVER ANYONE ASKS
ME TO MESS AROUND WITH FILESYSTEMS

Filesystem specific commands

- ext2/ext3/ext4
 - tune2fs
 - set options
 - manage quotas
 - resize2fs
 - shrink or expand
 - dumpe2fs
 - show filesystem data
- xfs
 - xfs_admin
 - set options
 - xfs_growfs
 - expand
 - xfs_info
 - show filesystem data
 - xfs_quota
 - manage quotas

Disk Partitions

- Subdivides a disk (or storage system)
 - Presents just like a disk to the file system
 - Simplifies backups
 - Provides a degree of protection
 - Difficult to change later
- Stored on the disk
 - MBR Master Boot Record
 - GPT Globally Unique ID Partition Table

Managing partitions

- Partition programs
 - fdisk (text UI)
 - text UI
 - parted
 - text UI
 - better for fixing stuff
 - gparted
 - GUI version
- DFIU
 - Make sure you are in the right disk!!!!!!!!!!
 - Moving partitions can destroy file systems
 - Expanding OK, shrinking typically not

Partition Table

- Partition properties
 - Primary/Secondary
 - Bootable flag
 - Start and end sectors
 - Size (sectors)
 - Type
- Partition types (hex)
 - 82 Linux swap
 - 83 Linux
 - 8e Linux LVM
 - a5 FreeBSD
 - fd Linux RAID
 - 86 NTFS

Machine A Partition Table

- **# fdisk -l /dev/sda**

```
Disk /dev/sda: 16 GiB, 17179869184 bytes, 33554432 sectors
Disk model: Virtual disk
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: gpt
Disk identifier: A7854EC3-3878-49A5-AFFF-3E7976B0F137
```

Device	Start	End	Sectors	Size	Type
/dev/sda1	2048	2099199	2097152	1G	EFI System
/dev/sda2	2099200	10487807	8388608	4G	Linux filesystem
/dev/sda3	10487808	29378559	18890752	9G	Linux LVM

- **# df -kh**

Filesystem	Size	Used	Avail	Use%	Mounted on
/dev/sda1	1022M	7.0M	1016M	1%	/boot/efi
/dev/sda2	4.0G	315M	3.7G	8%	/boot
/dev/mapper/r1-root	8.0G	1.4G	6.7G	17%	/
devtmpfs	4.0M	0	4.0M	0%	/dev
tmpfs	1.8G	0	1.8G	0%	/dev/shm
tmpfs	731M	27M	705M	4%	/run
tmpfs	366M	0	366M	0%	/run/user

RAID

- Redundant Array of Inexpensive Disks
 - Hardware chips or entire subsystems
 - Software RAID kernel module (`md`)
- RAID concepts
 - **Stripe** spread data across devices to improve throughput
 - **Mirror** save redundant information to survive drive failures

Types of RAID

- JBOD **J**ust a **B**unch **O**f **D**isks
- 0 Stripes data across 2 disks
- 1 Mirrors data to 2 disks
- 5 Stripe and mirror, survive 1 lost disk (min 3)
- 6 Stripe and mirror, survive 2 lost disks (min 4)
- 0+1 Mirror of stripes (4 disks)
- 1+0 Stripe of mirrors (4 disks)

RAID considerations

- RAID does not reduce the need for backups
- Hot spares are nice, but reduces capacity
- RAID5 many writes required hurt performance
- RAID5 vulnerable to desync on power fail
 - Battery backup improves robustness+performance
- Software RAID is inexpensive but slow

Linux Software RAID

- **md (multiple disks)**
- Devices are physical disks or partitions
- Managed by mdadm
- **Creating an array**

```
mdadm --create /dev/md/name --raid-devices=3  
          --level=5 /dev/sdf1 /dev/sdg1 /dev/sdh1
```

- **Making the array persistent**

```
mdadm --verbose --detail --scan>/etc/mdadm.conf
```

Logical Volume Manager (LVM)

- Physical Volume
 - Disk drives, partitions or RAID subsystems
- Volume Group
 - Collection of physical volumes
- Logical Volume
 - Logical device (partition within volume group)

LVM Pros and Cons

- Software RAID
 - More flexible, but slower than hardware
- Logical volumes can be readily resized
 - Readily add more physical volumes
 - More disks or partitions
 - Most filesystems can expand, some can shrink
- Snapshots

LVM Commands

- Physical Volume
 - pvcreate
 - pvdisplay
 - pvchange
 - pvck
 - pvremove
 - pvresize
- Volume Group
 - vgcreate
 - vgdisplay
 - vgchange
 - vgck
 - vgremove
 - vgextend
 - vgscan
- Logical Volume
 - lvcreate
 - lvdisplay
 - lvchange
 - lvscan
 - lvremove
 - lvextend

Simple LVM Recipe

- Label device(s)
 - `pvcreate /dev/sdb1`
- Create volume group
 - `vgcreate SA /dev/sdb1`
- Create logical volumes
 - `lvcreate -n tmp -L 1G SA`
 - `lvcreate -n mysql -L 4G SA`
- Make file systems
 - `mkfs -t xfs /dev/mapper/SA-tmp`
 - `mkfs -t xfs /dev/mapper/SA-mysql`
- Mount filesystems
 - `mount /dev/mapper/SA-tmp /tmp`
 - `mount /dev/mapper/SA-mysql /var/lib/mysql`

Swap

- Disk blocks that can be used to swap memory blocks to in virtual memory systems
 - Can be useful for large occasional memory hogs
 - Heavily using swap kills performance
- Linux swap commands
 - `mkswap` set up device for swap
 - `swapon` show or enable swap
 - `swapoff` disable swap

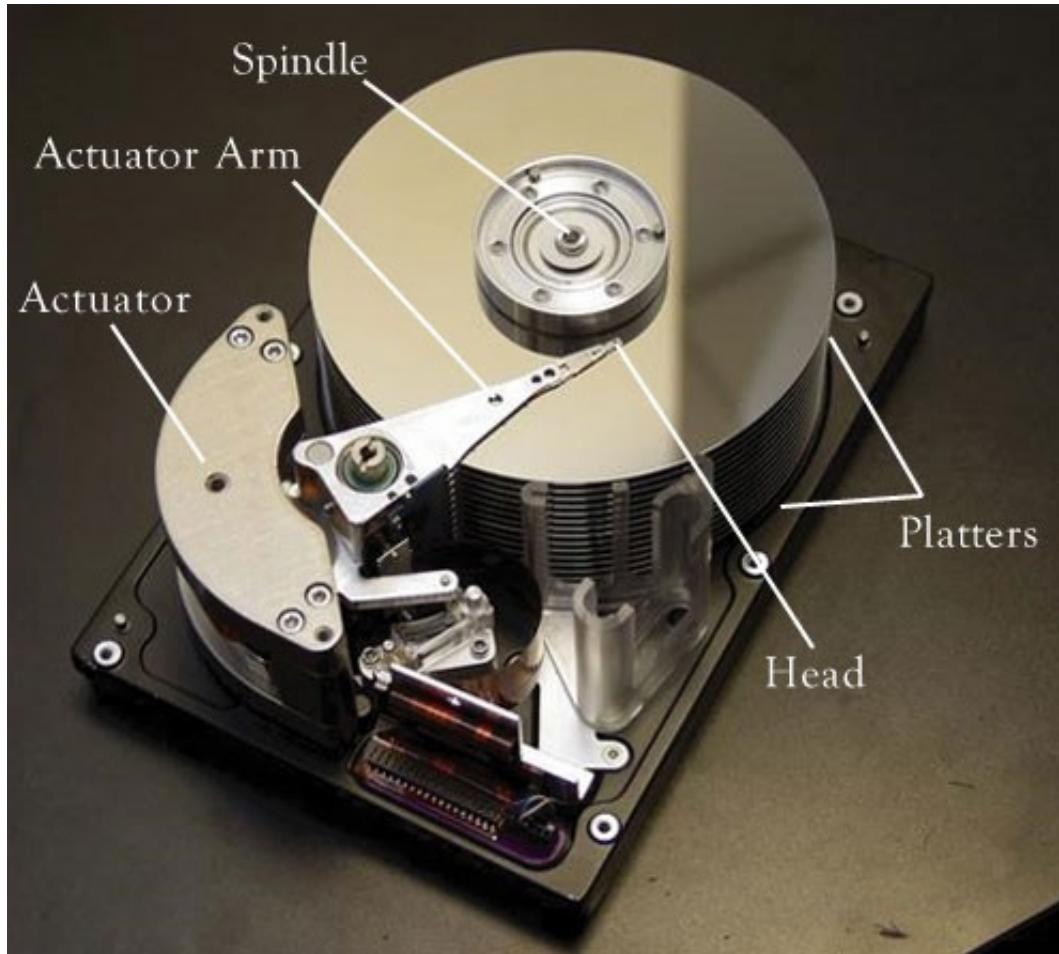
Network File System

- NFS is the native Unix Network File System
 - NFSv4 is the current versions
- Allows a server to export a subtree to clients
- Clients mount the subtree like a drive
 - `mount emc:/vol1 /home`
- Automount can mount or unmount on demand

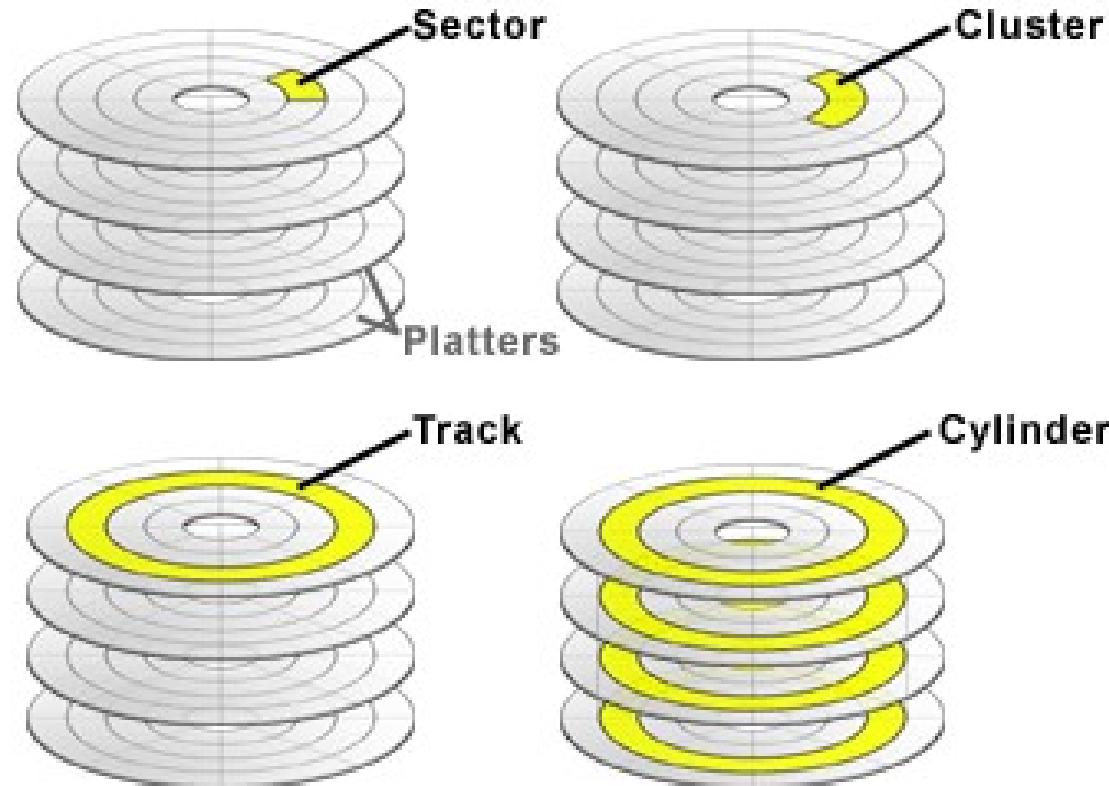
Disk drive types

- Mechanical
 - Slower 4-15 ms
 - Larger capacity
 - Less expensive
 - Sequential access must faster than random
 - Unlimited rewrites
 - Failure often predictable
- Solid State
 - Faster 0.1 ms
 - Lesser capacity
 - More expensive
 - Fast access regardless of pattern
 - Finite rewrites
 - Often fails without warning

Mechanical Hard Drives

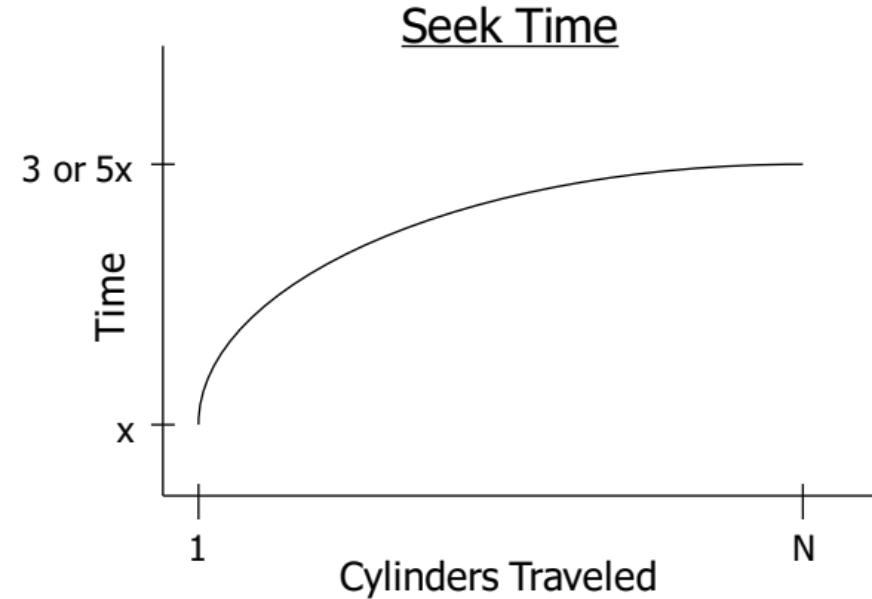


Mechanical Hard Drive Organization



Mechanical Hard Drive Properties

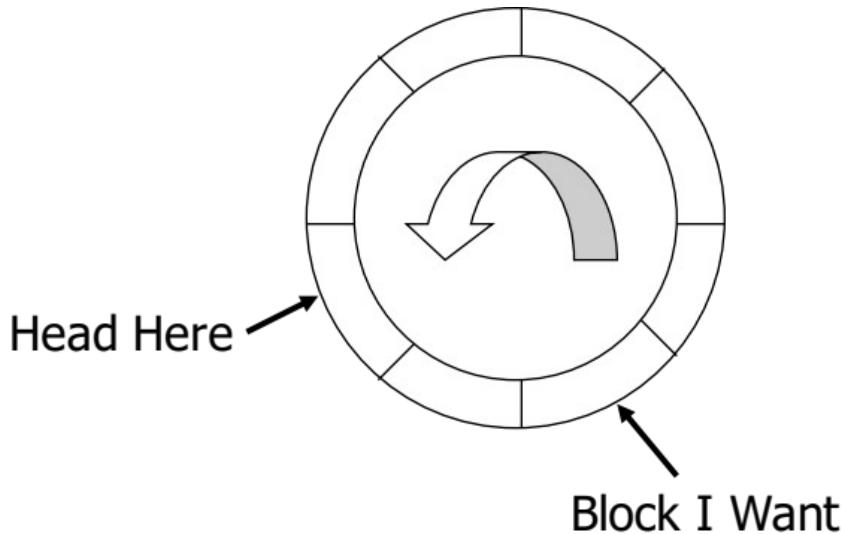
- Access time factors
 - Seek time
 - Acceleration
 - Transit
 - Deceleration
 - Settling
 - Rotational delay
 - Transfer time



Disk scheduling algorithms

- First come, first served
- Shortest Seek Time First
- SCAN (elevator algorithm)
- C-SCAN (one way elevator)
- C-LOOK (stops at first/last requested cylinder)

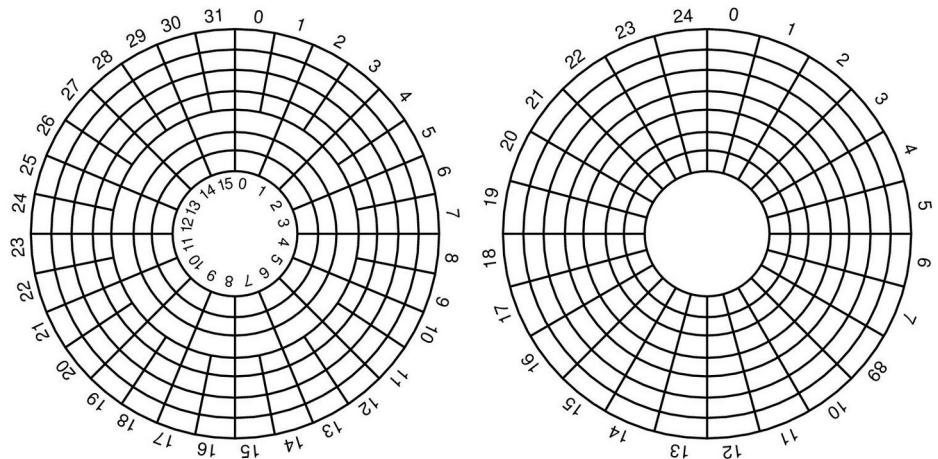
Rotational Delay



HDD Spindle [rpm]	Average rotational latency [ms]
4,200	7.14
5,400	5.56
7,200	4.17
10,000	3.00
15,000	2.00

Transfer Rate

- Transfer rate
 - Speed of media
 - Linear density
 - Faster on the outside
 - Potentially more sectors per track
 - Track total/rotation



Disk Interface Alphabet Soup

- Legacy Interfaces
 - IDE (ATA, PATA) *Integrated Drive Electronics*
 - SCSI *Small Computer System Interface*
- Current Interfaces
 - SATA *Serial ATA [1.5-6.0 Gb/s]*
 - SAS *Serial Attached SCSI [3-22.5 Gb/s]*
 - PCIe (NVMe) *Peripheral Component Interconnect Express*
 - USB *Universal Serial Bus*

Overall performance

- Access time = seek + rotation + transfer
- Write performance may be slower for SSD
 - Erase before write
- Caching
- Read-ahead
- RAID

Disk failures

Why you should have good backups

- MTBF *Mean Time Between Failure*
- S.M.A.R.T *Self-Monitoring, Analysis and Reporting Technology*
- Bad blocks, bad cells and self-healing
 - Low level formatting
- Drive types
 - Value - big and slow
 - Mass-market - fast and hot
 - NAS - robust and cool
 - Enterprise - fast and expensive

Logging

Linux System Administration
Fall 2023

Importance of Logging

- History of events
 - Kernel events
 - System reboots
 - System errors
 - Service start/stop
 - Startup warnings
 - cron jobs
- Audit trail
 - logins
 - refused logins
 - sudo commands
 - files accessed
 - invalid or refused
 - mail messages
 - DHCP requests

Logging types

- **syslog**
 - IETF standard
 - includes remote logging
 - facility.severity
- **systemd-journal**
 - logging by systemd
 - binary format
- Custom logging
 - Apache
 - access
 - error
 - sendmail
 - sudo

syslog

- rsyslog newer version
 - IETF standard
 - Systematic logging
time host [PID] message
- Configured by /etc/rsyslog.conf and /etc/rsyslog.d/*.conf
 - selectors facility.level action

syslog selectors

- facilities (*=all except mark)
 - auth authorization
 - authpriv sensitive auth
 - cron cron daemon
 - daemon system daemons
 - kern kernel
 - localX local messages
 - X = 0-7
 - mark regular time
 - mail mail
 - syslog internal
 - user default
- level (minimum) *=all
 - emerg system panic
 - alert action required
 - crit critical error
 - err error condition
 - warning could be a problem
 - notice pay attention
 - info informational
 - debug testing
 - none nothing

syslog actions

- `filename` Append to filename (- means no sync)
- `@hostname` Send to remote host
- `@ipaddress` Forward to IP port UDP 514
- `@@ipaddress` Forward to IP port TCP 514
- `| fifo` Write to named pipe
- `user1, user, ...` Write message to user consoles
- `*` Write to all user consoles
- `^prog;template` Send to program
- `~` Discard

syslog example (Machine C)

```
auth,authpriv.*          /var/log/auth.log
*.*;auth,authpriv.none  -/var/log/syslog
cron.*                  /var/log/cron.log
daemon.*                -/var/log/daemon.log
kern.*                  -/var/log/kern.log
user.*                  -/var/log/user.log

# Split mail
mail.*                  -/var/log/mail.log
mail.info               -/var/log/mail.info
mail.warn               -/var/log/mail.warn
mail.err                /var/log/mail.err

# Catchall
*=info;*=notice;*=warn;auth,authpriv.none; \
cron,daemon.none;mail.none  -/var/log/messages
```

systemd-journal

- systemd attempt to take over logging
- Binary journal for quick retrieval
 - More difficult to manage than text files
- Access with `journalctl`
 - See output in `systemctl status`
- Works in tandem with syslog
 - Log messages also forwarded to syslog

journalctl examples

- Show all boots

- `journalctl --list-boots`

- Show logs as they arrive (follow)

- `journalctl -f`

- Show entries for a service (`sshd`)

- `journalctl -u sshd` (**RedHat**)

- `journalctl -u ssh` (**Debian**)

Other logging

- Apache
 - Configured in <virtualHost>
 - TransferLog - standard log format
 - CustomLog - custom fields and order
 - Stored in /var/log/httpd or /var/log/apache2
- sudo
 - Configured in /etc/sudo_logsrvd.conf
 - Stored in /var/log/secure or /var/log/auth.log

logger

- Program to add syslog entries (from scripts)
- `logger -p facility.level message`
 - `logger -p daemon.info "Start foo"`
 - `logger -n log.dm.com "Machine C boot"`
- Also useful for testing syslog configs
- systemd-journal has to be different
 - `echo 'Start foo' | systemd-cat -t myapp -p info`

Managing logs

- Logs are stored in `/var/log` or `/var/log/*`
- Log aging set by site policy
- `logrotate`
 - `/etc/logrotate.conf` and `/etc/logrotate.d/*`
 - Starts new log file daily, weekly, monthly, etc.
 - Keeps older logs as `.1`, `.2.gz`, `.3.gz`, ...
- By default `systemd` journal grow without limit
 - Limit in `/etc/systemd/journald.conf` with `SystemMaxUse`

Central Logging

- It doesn't take much to drown in logs
 - Roll your own with syslog and scripts
 - ELK : Elasticsearch, Logstash, Kibana
 - Network Management Software (e.g. Observium)
- Make sure all your systems agree on time
 - NTP to synchronize clocks
 - Consistent time zone