COMP09024 Unix System Administration

Lecture 8: System Configuration and Monitoring

Duncan Thomson/Hector Marco

UWS

Trimester 1 2020/21

Outline

8.1 Managing Unix Systems

Command Line Interface

- Unix servers were originally command line only, and so configuration was performed on the command line
- Although most Unix systems now include a GUI (usually based on X11), command-line text-based administration has persisted
- Advantages include:
  - Existing tools used to perform configuration
  - Low overhead and impact on other software
  - Remote administration identical and low bandwidth
- Disadvantages:
  - System adminstrators need to be comfortable with CLI and working with text files
- . . . but hopefully by now, you already are!?

Configuration File

- Configuration files are found in `/etc/` (eg: `/etc/passwd`, `/etc/hostname`, `/etc/inittab` ...
- In many cases, subdirectories of `/etc/` are used for more complex subsystems (eg `/etc/apt/`, `/etc/cups/`, `/etc/network/`
- In (almost) all cases, configuration files are text-based
- These may be documented in three ways:
  - A `man` page (in section 5)
  - Comments in the file itself
  - Example configurations (sometimes included in the comments)
- Since they are text-based, text editors (like `vi`) can be used to make changes

Servers and Daemons

- Many services on Unix run without being connected to a terminal
- Such server processes are often known as *daemons*
- Such processes need to:
  - Be automatically run at boot time (if required)
  - Be able to be stopped and started as required
  - Be able to be told to re-read their configuration files if they are changed
  - Be monitored and restarted if they stop unexpectedly
- The `init` system typically provides mechanisms for some or all of these

Secure Shell

- If systems are administered remotely, a mechanism to access the command line remotely is required
- In days gone by, `telnet` or the `rlogin`:
  - Authentication but not Encryption (data unencrypted)
- Modern Unix systems provide Secure Shell (SSH):
  - Authentication and Encryption
- A secure shell server is run on the server (`sshd`).
- The `ssh` command will connect to a remote machine running SSH server, and once credentials have been checked command line access is granted

```
ssh   servername
ssh   username@servername
```

Secure Shell Capabilities

- SSH client checks server key on connection, and refuses to connect if this has changed (to prevent man-in-the-middle attack).
- SSH can copy files to/from the remote server (with `scp`):

```
scp  localfile  user@server:remotefile
scp  user@server:dir/remotefile  localfile
```

- SSH provides key management to automate authentication
- SSH can tunnel X11 (GUI) data with the `-X` flag
- SSH can tunnel TCP connections through the connection:
  - `-R` to tunnel from remote port to local port
  - `-L` to tunnel from local port to remote port

8.2 The vi Text Editor

vi Revisited

- vi is the most ubiquitous text editor on Unix
- Also the basis for sed the stream editor
- . . . and is also quick to use, once you've learned the basics
- Remember, vi is modeful:
  - In command mode (or 'normal' mode), every keystroke is a (case-sensitive) command
  - In insert mode, you are typing into the file (exit with ESC)
  - In 'bottom line mode', commands from the legacy ex editor can be used (including file save, quit, etc)
- It's also *case-sensitive* (upper and lower case commands are often related, but not the same)
- All commands can be preceded by a number, which in most cases repeats the command that number of times

Movement Commands

- `hjkl` — movement left, down, up, right (or use arrows)
- `w`, `b`, `e`, `W`, `B`, `E` — move forwards, backwards or to end of a word (including punctuation with capitals)
- `0` and `$` — move to start or end of line
- `H`, `M` and `L` — move to top, middle or bottom of screen
- Ctrl-`B` and Ctrl-`F` — move back / forward a screenful
- Ctrl-`U` and Ctrl-`D` — move back / forward half-screenful
- `f`*c* or `F`*c* — move onto next/previous occurrence of char
- `t`*c* or `T`*c* — move up to (but not including) next/previous occurrence of character
- *n*`G` — go to a given line number (or to end of file without *n*)
- `%` — go to matching opening or closing parenthesis
- `/`*regexp* or `?`*regexp* — search forward / backward

Editing Commands

- a or A — append after cursor / end of line (insert mode)
- i or I — insert before cursor / start of line (insert mode)
- o or O — open new line after/before current one (insert)
- d*move* or y*move* — delete or copy (yank) specified text
- c*move* — change specified text (insert mode)
- D or C — delete or change to end of line (insert mode)
- dd, yy or cc — delete, yank or change whole line
- p or P paste deleted/yanked content after / before current position
- x or X — delete character forwards / backwards
- r*c* — replace current character
- ~ — change case of current character
- J — joins current and next line

Other Commands

- `u` — undo last command
- `.` — repeat last editing command
- `n` or `N` — repeat last regexp search (same / reverse dir)
- `:w` or `w` *`filename`* — save (write) file
- `:q` or `q!` quit / quit without saving
- `:wq` — save and quit
- `:r` *`filename`* — read in a file at cursor position
- `:s/`*`regexp`*`/`*`string`*`/` — search and replace first occurrence on current line, or precede with a line range:
  - `10,20` for lines 10–20
  - `1,$` for all lines
  - `.,+10` for current line to 10 lines later
  - Suffixed `g` replaces all occurrences on the line(s)

Example Commands

- The true power of `vi` comes from:
  - Combining edit and movement commands
  - Preceding commands with numbers
  - Using the 'repeating' commands
- Examples:
  - `dG` — delete from here to the end of the file
  - `10dd` — delete 10 lines (which can be pasted back with `p` elsewhere)
  - `20~` — change case of 20 characters
  - `c3w` — change the following 3 words into inserted text
  - `d%` — delete contents of this parenthesis

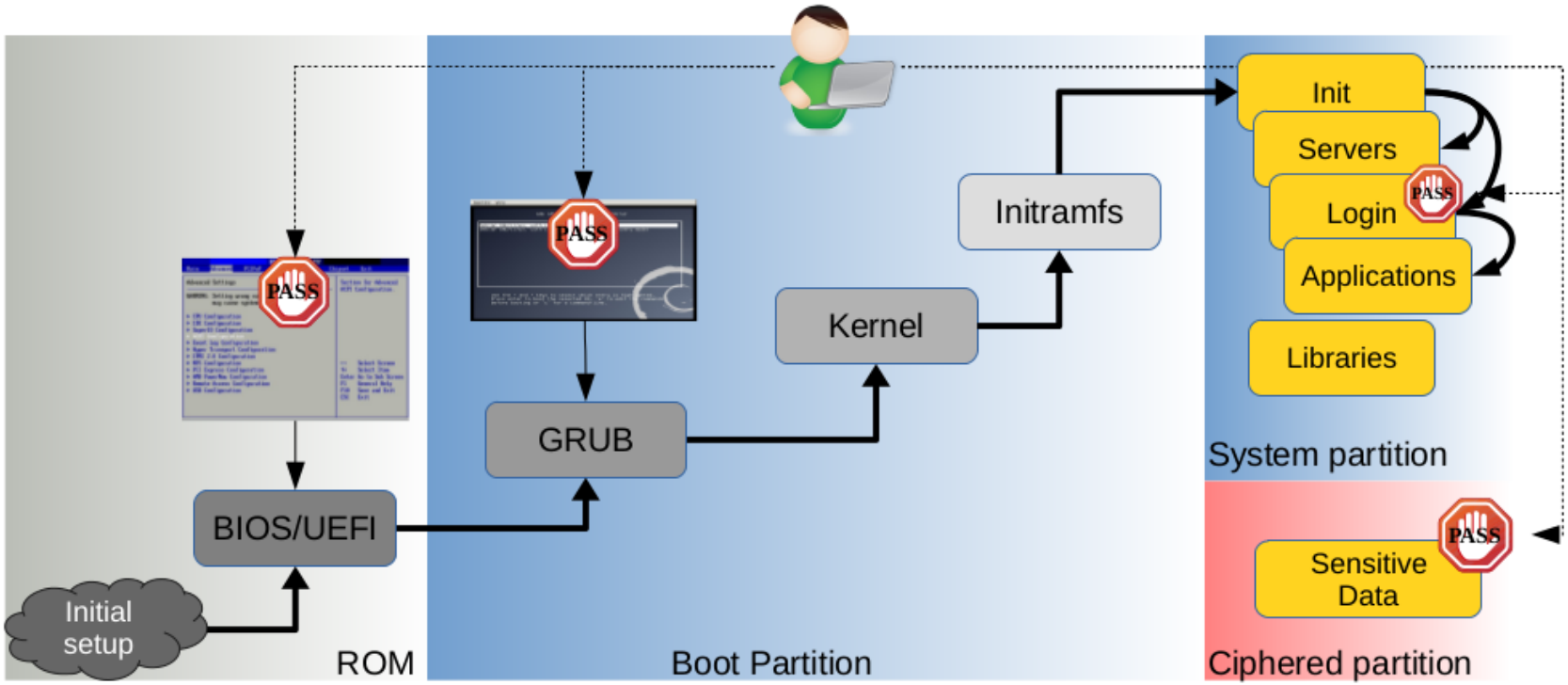
8.3 The Boot Process

Overview of System Boot (1/2)

When a system boots it goes through a number of stages:

1. Load and run initial firmware (eg BIOS, UEFI, etc)
   - Perform power-on self-test (POST)
   - Detect attached devices
   - Locate boot loader (eg on master boot record (MBR))
2. Load and run boot loader software (eg GRUB, LILO, etc)
   - Read devices and filesystems
   - Locate kernel and any other initial filesystem
3. Load and run OS kernel
   - In some cases an initial filesystem may also be required
4. Mount main root filesystem at `/`
5. Run first process (`init`)
6. Bring up other filesystems and processes as required (under control of `init`)

Overview of System Boot (2/2)



Boot system Overview. User's HOME partition ciphered.

(source: http://hmarco.org/bugs/CVE-2015-8370-Grub2-authentication-bypass.html)

Boot Loaders

- A range of boot loaders are available for Unix systems
- Some are tied strongly to one operating system, e.g:
  - LInux LOader (LILO)
  - `isl` for HP-UX
- GNU Grand Unified Bootloader (GRUB) is a modern modular boot loader used by multiple Unix variants, including Solaris and Linux
  - Configured from script in `/boot/grub/grub.cfg`
  - Can load modules for filesystems, etc
  - Allows selection between multiple boot options, or manual boot

Service managers

System-V, Upstart and systemd:

- The classical “System-V” init service.
  - An obsolete way to start and stop groups of services.
  - It uses the runlevel concept.
- Around 2006, the “Upstart” init service replaced SysV.
  - It replaces “System-V”.
  - No more `/etc/inittab` file.
- Nowadays, most Linux distributions use “systemd”:
  - The runlevel concept is obsolete.
  - It provides a mapping between runlevels and systemd targets (e.g: runlevel 5 $\rightarrow$ `graphical.target`)

System V `init`

- The traditional Unix `init` system is System V `init`
- This defines a number of runlevels:
  - Runlevel 0 is for shutting down the system
  - Runlevel s is for single-user mode (for maintenance)
  - Runlevel 6 is normally for rebooting
  - Runlevels 1–5 can be manually configured
- Overall behaviour of `init` controlled by `/etc/inittab`
- Scripts used to start/stop services when changing runlevel
- The `telinit` command can switch between runlevels
- Other commands provide shortcuts for certain runlevels:
  - `reboot` reboots the system immediately (runlevel 6)
  - `halt` shuts down the system immediately (runlevel 0)
  - `shutdown` allows delayed halt (or reboot with `-r`) along with a message for users

/etc/inittab

- /etc/inittab has four colon-separated fields
- ID field identifies entry with up to four characters
  - Some IDs are traditionally used for certain entries, eg id for initial default runlevel; numbers relating to login processes on ttys; etc
- Runlevels field lists runlevels in which service should be active
- Action specifies how to run process, eg:
  - initdefault specifies default runlevel
  - sysinit and boot run on boot only
  - once and wait run once on entering runlevel
  - respawn restarts on exit
  - ctrlaltdel runs when Ctrl-Alt-Del pressed
- Process specifies command (and arguments) to run

Example /etc/inittab

```
# default runlevel is 2
id:2:initdefault:

# system initialisation on boot
si::sysinit:/etc/init.d/rcS

# single-user mode
~~:S:wait:/sbin/sulogin

# rc script for each runlevel
l0:0:wait:/etc/init.d/rc 0
l1:1:wait:/etc/init.d/rc 1
l2:2:wait:/etc/init.d/rc 2
l3:3:wait:/etc/init.d/rc 3
l4:4:wait:/etc/init.d/rc 4
l5:5:wait:/etc/init.d/rc 5
l6:6:wait:/etc/init.d/rc 6
```

```
# what to do what Ctrl-Alt-Del
ca:12345:ctrlaltdel:/sbin/shutdown \
                    -t1 -a -r now

# what to do when power fails
pf::powerwait:/etc/init.d/powerfail \
                     start
pn::powerfailnow:/etc/init.d/powerfai
                     now
po::powerokwait:/etc/init.d/powerfail
                     stop

# login prompts on virtual terminals
1:2345:respawn:/sbin/getty 38400 tty1
2:23:respawn:/sbin/getty 38400 tty2
3:23:respawn:/sbin/getty 38400 tty3
4:23:respawn:/sbin/getty 38400 tty4
```

init Scripts

- The rc command (called in the previous example) runs so-called init scripts for various services
- The init scripts are stored in /etc/init.d/ or similar, eg /etc/init.d/cron
- Each script can take various parameters
  - The start parameter starts the service
  - The stop parameter stops the service
  - restart and/or reload parameters may also work
- The service command can often be used instead
- For example, to stop the cron service, can use either:

```
/etc/init.d/cron stop
service cron stop
```

Systemd init

- It is a replacement of the UNIX System V.
- Unification of basic Linux configurations and service behaviors across all Linux distributions.
- Instead of /etc/inittab we have a /etc/systemd/system/ directory. This directory holds symlinks to /lib/systemd/system/ which contains init scripts.
- Unlike other init systems, you do not have to know a scripting language to interpret the init files used to boot services or the system.
- systemctl [start,stop,restart,reload,status] [name.service]
  - systemctl status boot.mount

Systemd init

CUPS status example (an open source printing system):

```
$ systemctl status cups.service
```

```
hecmargi@macal:system$ systemctl status cups.service
● cups.service - CUPS Scheduler
   Loaded: loaded (/lib/systemd/system/cups.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2018-10-17 17:48:54 BST; 57min ago
     Docs: man:cupsd(8)
 Main PID: 1193 (cupsd)
    Tasks: 1 (limit: 4915)
   CGroup: /system.slice/cups.service
           └─1193 /usr/sbin/cupsd -l

Oct 17 17:48:54 macal systemd[1]: Started CUPS Scheduler.
```

Systemd services:

```
$ ls -l /lib/systemd/system/*.service
-rw-r-r- ...  /lib/systemd/system/accounts-daemon.service
-rw-r-r- ...  /lib/systemd/system/acpid.service
-rw-r-r- ...  /lib/systemd/system/alsa-restore.service
-rw-r-r- ...  /lib/systemd/system/alsa-state.service

...      ...  ...
```

Systemd - Linux distributions

Linux distribution	Date added to software repository[a]	Enabled by default?
Alpine Linux	N/A (not in repository)	No
Android	N/A (not in repository)	No
Arch Linux	January 2012[52]	Yes
CentOS	April 2014	Yes
CoreOS	July 2013	Yes
Debian	April 2012[57]	Yes
Fedora	November 2010 (v14)[60]	Yes
Gentoo Linux[b]	July 2011[61][63][64]	No
Knoppix	N/A	No [65][66]
Mageia	January 2011 (v1.0)[67]	Yes
Mint	June 2016 (v18.0)	Yes
openSUSE	March 2011 (v11.4)[69]	Yes
Red Hat Enterprise Linux	June 2014 (v7.0)[71]	Yes
Slackware	N/A (not in repository)	No
Solus	N/A	Yes
SUSE Linux Enterprise Server	October 2014 (v12)	Yes
Ubuntu	April 2013 (v13.04)	Yes
Void Linux	June 2011, removed June 2015 [73]	No

/etc/rcN.d/

- The particular init scripts started (and stopped) in particular runlevels are controlled by the contents of the /etc/rcN.d/ directories
- These directories (one per runlevel) contain symbolic links
- Each symbolic link links to an init script in /etc/init.d/
- Each link is named *LNNservice* (*L* is a letter, *NN* is a number), and the name determines how and when each script is run:
  - Links beginning with K (for kill) are run with stop parameter
  - Links beginning with S are run with the start parameter
  - Scripts are run in order of the number *NN*

8.4 Controlling Services and Systems

Using `init` Scripts and `service`

- If System V `init` is used, the canonical way to control services is via the `init` script:
  - `/etc/init.d/`*`servicename`* `start` to start a service
  - `/etc/init.d/`*`servicename`* `stop` to stop a service
  - `/etc/init.d/`*`servicename`* `restart` to restart a service
  - `/etc/init.d/`*`servicename`* `reload` to reload configuration for a service
- Usually the `service` command provides another way of accessing this functionality:
  - `service servicename start` to start a service
  - `service servicename start` to stop a service
  - `service servicename reload` to reload config
  - . . . and so on

Special Cases

- Some services provide their own commands to start, stop and otherwise control operation, examples include:
  - The Apache webserver (`apachectl` or `apache2ctl` commands)
  - The BIND DNS server (the `rndc` command)
- These are usually additional to System V `init` scripts, rather than instead of them
- Other `init` systems provide their own commands:
  - `systemd` uses `systemctl`
  - `launchd` uses `launchctl`
  - Upstart uses `initctl`

Kernel Configuration I

Most Unix systems have at least two mechanisms for configuring how the kernel operates:

1. Add kernel features:
   - Load new modules to the running kernel (e.g. drivers)
   - In Linux, all modules end with the `.ko` extension
   - Modules can be found in `/lib/modules`
   - Linux version 5 modules are in `/lib/modules/5.0.0-23-generic/kernel/`
   - `modprobe ntfs` adds NTFS file system support
   - `lsmod` to list currently loaded modules

```
user@debian~$ lsmod
Module                  Size  Used by
ntfs                  106496  0
pci_stub               16384  1
...                      ...  .
```

Kernel Configuration II

2. Configure kernel options
   - Controlling the run-time behaviour of the kernel
   - `sysctl -a` lists all `sysctl` settings
   - `sysctl` *`sys.ctl.name`* prints individual setting
   - `sysctl` *`sys.ctl.name = value`* sets a value
   - `/proc/sys/kernel` contains files controlling a range of kernel parameters

Example: Obtain the current Address Space Layout Randomization (ASLR) mode:

```
user@debian~$ sysctl kernel.randomize_va_space
kernel.randomize_va_space = 2
user@debian~$ cat /proc/sys/kernel/randomize_va_space
2
```

- `0` ASLR is off
- `1` ASLR is on but HEAP is not randomized
- `2` ASLR is on and HEAP is randomized

8.5 Logging, Monitoring and Auditing

Logging and Log Files

- When a system or service is not running correctly, it can be difficult to diagnose the problem using only the `stderr` command-line output
- Most services can produce detailed log messages
- These are typically stored in text-based 'log files', usually found under `/var/log/`, eg:
  - `/var/log/syslog` for general messages
  - `/var/log/auth` for authentication logs
  - `/var/log/wtmp` contains user login records (not text)
  - `/var/log/apache2/access.log` for web access logs
- Some services write directly to log files, others use the `syslog` system to unify and control their log messages

syslog

- The syslog system — implemented by (eg) rsyslogd or Syslog-NG — provides a unified logging framework
- Although origins lie in Unix, it is used on many systems
- Every log message contains various pieces of information:
  - Facility (kern, user, mail, auth, daemon, local0–local7, . . .)
  - Severity (emerg, alert, crit, error, warn, notice, info, debug)
  - A timestamp
  - A host name or address
  - A tag (usually the name of program generating the entry)
  - A (human-readable) message
- The logger command can send a log message manually
- All messages are sent to syslog process, which decides how to deal with each: ignore, append to a file, send over the network, alert one or more users, etc

Example /etc/rsyslogd.conf File

```
# all security stuff to auth.log
auth.*                  /var/log/auth.log
# everything except security to syslog file
*.*;auth.none           /var/log/syslog
# all daemon stuff to daemon.log
daemon.*                /var/log/daemon.log
# mail messages to separate mail log file
mail.warn               /var/log/mail.log
# debug messages only to debug log
*.=debug                /var/log/debug
# critical or emerg messages to everyone logged in
#               and by UDP to remote server at 10.0.0.10
*.crit                  * @10.0.0.10
```

Monitoring Systems

- One method is to read log files — tedious and inefficient
- `tail -f` can keep track of recent additions to a single file
- When searching for specific information, Unix's filters (such as `grep`, `cut`, `tail` and others) can be very useful,
- There are also tools such as:
  - `ps` and `top` for monitoring processes
  - `who` and `w` for monitoring users
  - `df` and `free` for monitoring filesystem and memory usage
  - `netstat` for network statistics and connections
- Many kernels have additional instrumentation allowing fine-grained information on system performance, eg:
  - `vmstat` on Linux
  - `sysstat` package with `iostat`, `mpstat`, ...

Other Monitoring Tools

- SNMP: Simple Network Management Tool
  - Allows network access to system performance data (including CPU, memory, disk space, network statistics) from a network management station
  - Requires installing an SNMP *agent* on the managed system
  - Basic tools available for free, or can form part of large network-wide systems (OpenNMS, Tivoli, OpenView, eHealth)
- MRTG (Multi-Router Traffic Grapher)
  - Can provide graphs of performace data over varying time periods
  - Originally designed for network activity, but extends to other system data
- Nagios is another system/network monitoring tool (including SNMP data and other services)

Auditing Systems

- May sometimes require to audit systems for security — a variety of tools are available
- `nmap` (as seen in *The Matrix*) is a network port scanner with OS fingerprinting and service/software identification
- `tripwire` allows auditing of sensitive system files by creating cryptographic checksums and checking for changes
- Linux includes the 'audit' system which allows detailed monitoring of specified events relating to any system calls, such as:
  - File access
  - Network activity
  - User access

Summary

- Typical Unix system management: remote CLI, text files
- Secure Shell for remote access and more
- Using the `vi` editor
- Boot process: boot loaders
- The `init` process (PID 1)
- Service mangers: System-V, Upstart and systemd
- Kernel configuration: modules and sysctls
- Logging and log files
- The `syslog` system: severities, facilities
- Monitoring tools and systems: processes, users, memory; MRTG, SNMP
- Auditing tools: `nmap`, `tripwire`, the Audit system