TNE30019/TNE80014 – Unix for Telecommunications

System Bootup

Dr. Jason But

Swinburne University

Outline

- How Unix System Starts Up
- What Happens after System Boots
- Startup Scripts
- The `login` Process
- The shell

System Startup – UEFI/BIOS

- Initial OS to manage hardware (UEFI/BIOS)
- Firmware on motherboard
- Loaded after turning on computer
- Has basic functionality to access hardware
- Knows bootable hardware devices
  - User can define device order in setup
- Knows how to hand over to hardware for booting
  - Can start boot loader from hard disk
  - Can start PXE stack on network card

System Startup – Boot Loader

- How does an OS load?
- **Boot Loader** is small program that loads and runs OS kernel
- Needs to knows where kernel is stored on disk
- Often provides some means to
  - Select which kernel to boot
  - Set parameters for kernel to boot

Boot Loader Functions

1. Loads kernel executable
2. Starts kernel

System Startup – Kernel

- Is the actual OS

Kernel Startup Functions

1. Run system checks, e.g memory test
2. Setup memory management, CPU, interrupt handling
3. Detect hardware devices
4. Load initial RAM disk (Linux)
5. Load device drivers
   - Drivers create devices in `/dev`
   - Configures hardware ready for use
6. Start idle process, scheduler and other key kernel processes (**PID=0**)
7. Mount root file system
8. Launch `init` – process initialisation daemon (**PID=1**)

System Startup – Run Levels

- Unix has concept of booting into different run-levels
- Different start-up scripts/services are run depending on chosen run-level
- Eight runlevels, three of which are "standard"

Standard and common run-levels

0 Halt (standard)
1 Single-user mode (standard)
2 Local multi-user console (common)
3 Multi-user console (common)
5 Multi-user graphical desktop (common)
6 Reboot (standard)

System Startup – `init`

- `init` is first process and manages startup of system processes
- `/etc/inittab` is the configuration file for init
  - Specifies default run-level
  - If no default run-level `init` will prompt user
- `init` launches `rc` with run-level as parameter
- `init` starts virtual terminals
- On Linux, big push to replace `init` with `systemd`

`rc` script

Shell script to start basic services, daemons and applications

Linux `/etc/init.d/rc`
FreeBSD `/etc/rc`

System Startup – `rc`

- This is where you can customise system
- Script decides what processes to launch

FreeBSD

- Runs scripts in `/etc/rc.d` with the `start` flag
  - Example: `/etc/rc.d/ntpd start` – Launches ntp daemon
- Scripts written so processes are launched in correct order
- Variables configured in `/etc/rc.conf` tell scripts in `/etc/rc.d` whether to actually launch that process and command line parameters to use

Linux

- Directories maintained for each run level – location platform specific
- Each directory contains sym links to scripts to execute

System Shutdown

- Signal is sent to `init` to terminate
- On FreeBSD `init` runs `/etc/rc.shutdown`
  - Script tries to stop all processes launched by `rc`
- On Linux `init` executes scripts in `rc0.d` or `rc6.d`

`init` – Final Shutdown

After all services stopped

- Terminate any remaining processes
- Flush disks
- Terminate and shuts down system

Your system is booted ... and then

User Interaction

- Console: keyboard + mouse + display
- On servers user interaction is typically done via network connections (remote console)
- On clients need console for user interaction
- Console access on servers is also useful in emergencies

Unix systems output to **console**

- Traditionally terminal connected to serial port
- Nowadays console is tied to virtual terminal (`/dev/ttyv0`)
- Graphics card driver ties this terminal to display

The **login** Process

- Virtual terminals started by `init` will launch login application
- This is runnable process (`/bin/login`)

`login` process

- Displays the "`login:`" prompt
- Prompts for username and password
- Checks username and password against database
- If user is properly authenticated, grants access

`login` is **NOT** user interface

- Sets environment variables
- Launches yet another process – the **shell**
- Shell provides user interface

The Shell

- Shell is basically command interpreter
  - Prompt user for input
  - Execute user's commands
  - Display output
- Which shell login starts defined in user database

Many different shells with different functionality

`sh`	Bourne shell
`bash`	Bourne-again shell
`csh`	C shell
`tcsh`	TENEX C shell
`ksh`	Korne shell
`zsh`	Z shell

We Can Start Working Now