CSI3660 – System Administration

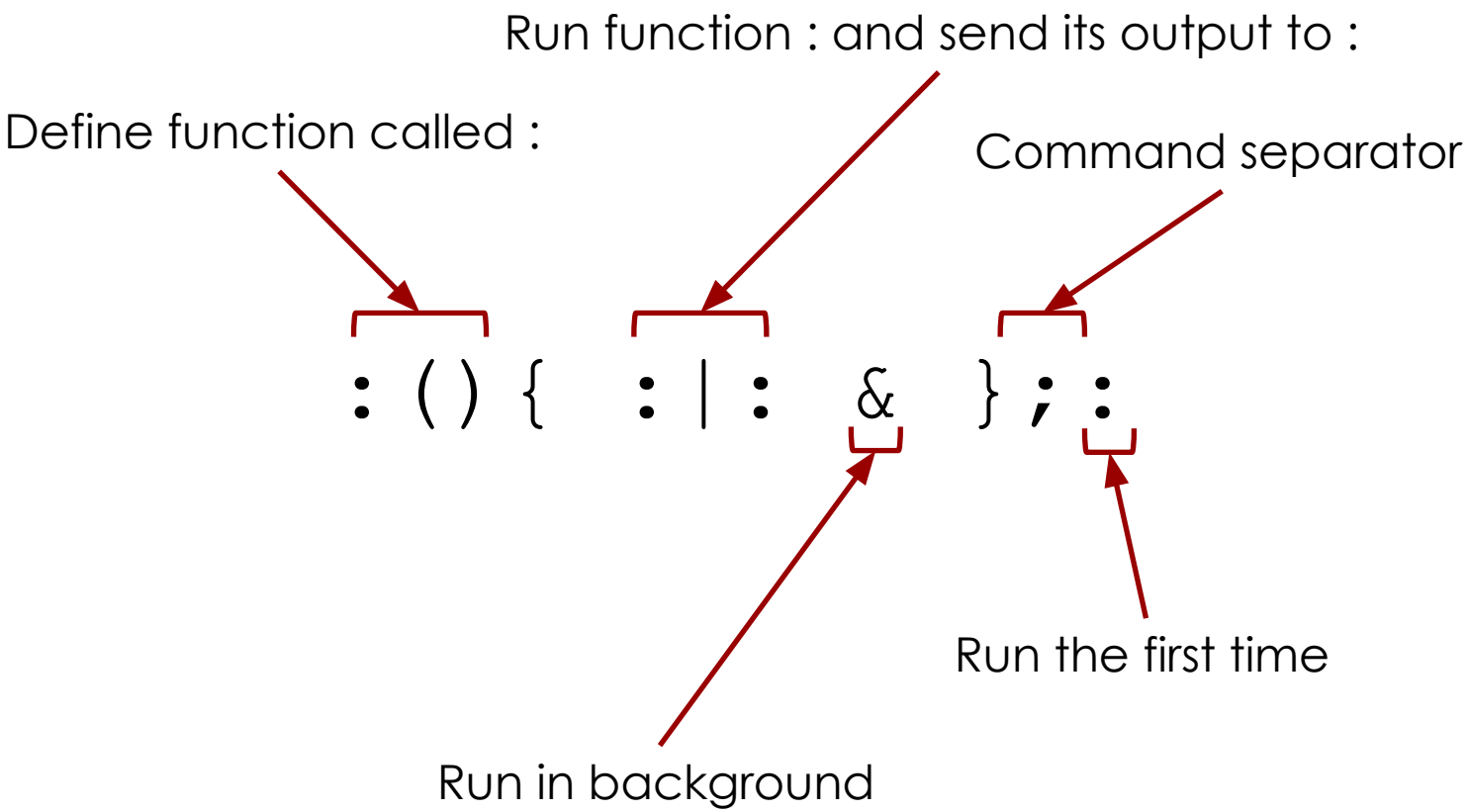
Prof. Fredericks

Process Management and Booting

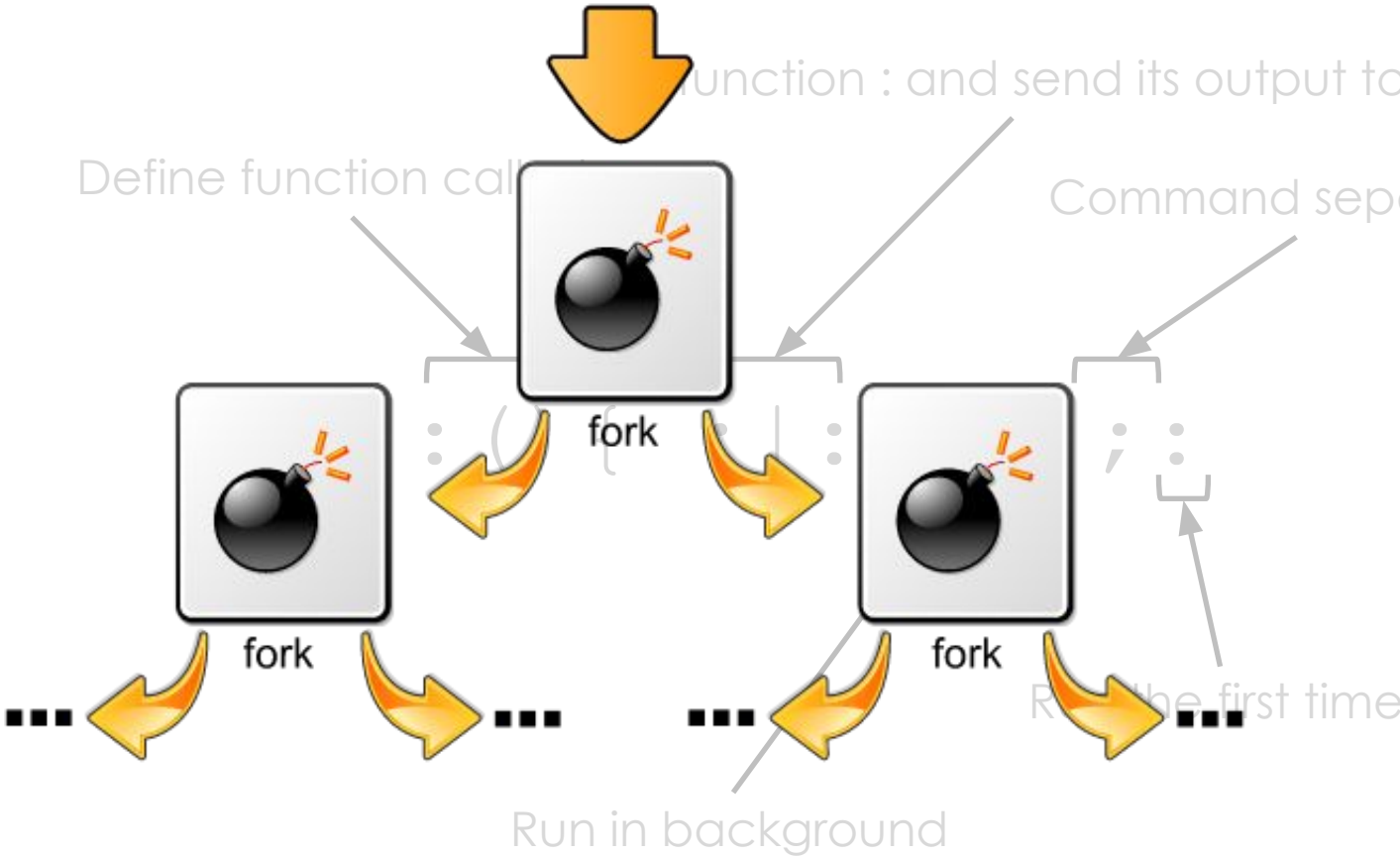
Outline

- Process management
  - Categorize different types of processes
  - View processes
  - Kill signals
  - Execute binary programs / shell scripts
- Startup sequence
  - Different types of system initializations (different UNIX flavors)
  - Details about different levels of system execution
  - X Windows

What is this?


Run function : and send its output to :
Define function called :
Command separator
: ( ) { : | : & } ; :
Run the first time
Run in background

What is this?


unction : and send its output to :
Define function cal
Command separator
fork
fork
fork
Run in background

:(

```
$ ls
$ ls
bash: no more processes:
      no more processes
```

In-class assignment/discussion

- Write down 2 ways to deal with this problem

Make SSH Start on Boot

Apparently installing this no longer auto-starts when you reboot

(It used to in Scientific 6)

```
sudo systemctl status sshd.service

sudo systemctl enable sshd.service

sudo systemctl status sshd.service
```

Linux Processes

- Program
  - Structured set of commands stored in an executable file (binary file)
  - Executed to create a process
- Process
  - Program running in memory and on CPU
- User process
  - Process begun by a user on a terminal
- Daemon process
  - System process
  - Not associated with a terminal

Linux Processes

- Process ID (PID)
  - Unique identifier assigned to a process
- Child process
  - Process started by another process (parent process)
- Parent process
  - Process that has started other processes (child processes)
- Parent Process ID (PPID)
  - PID of the parent process
- The init daemon has a PID of 1 and a PPID of 0
  - 0 refers to the kernel itself

Linux Processes

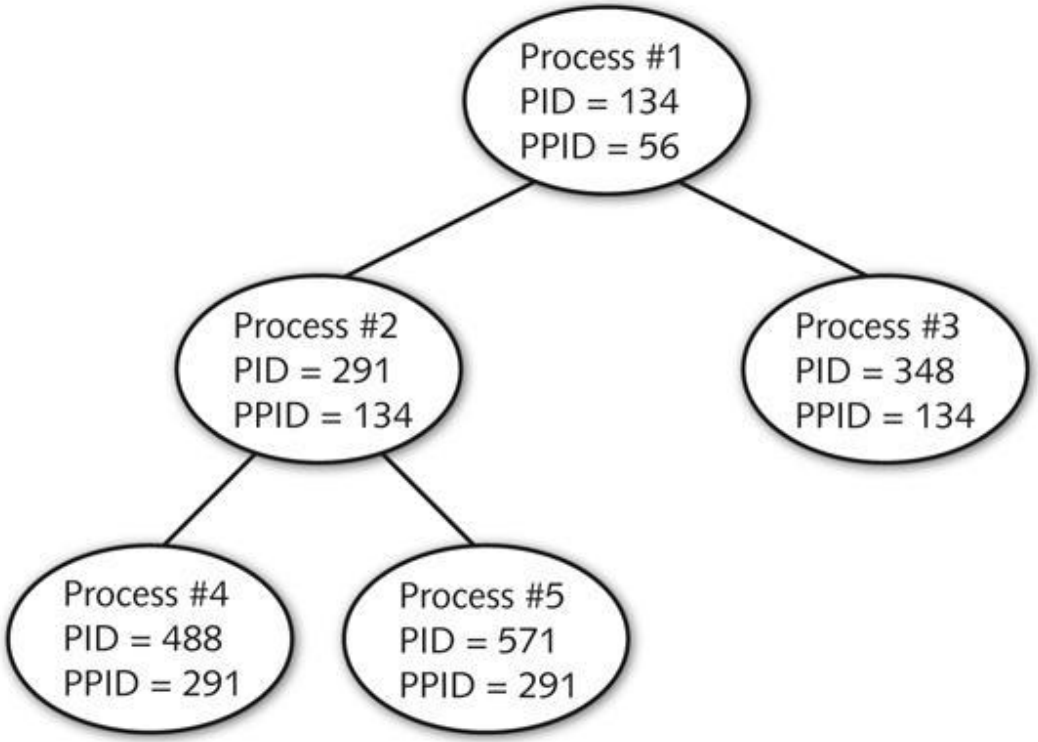


Figure 9-1: Parent and child processes

Linux Processes

- init daemon
  - Starts most other daemons during the system initialization process
  - Including those that allow for user logins
- The login program starts a BASH shell
  - BASH shell then interprets user commands and starts all user processes
- Each process on the Linux system can be traced back to the init daemon by examining PPIDs

Linux Processes

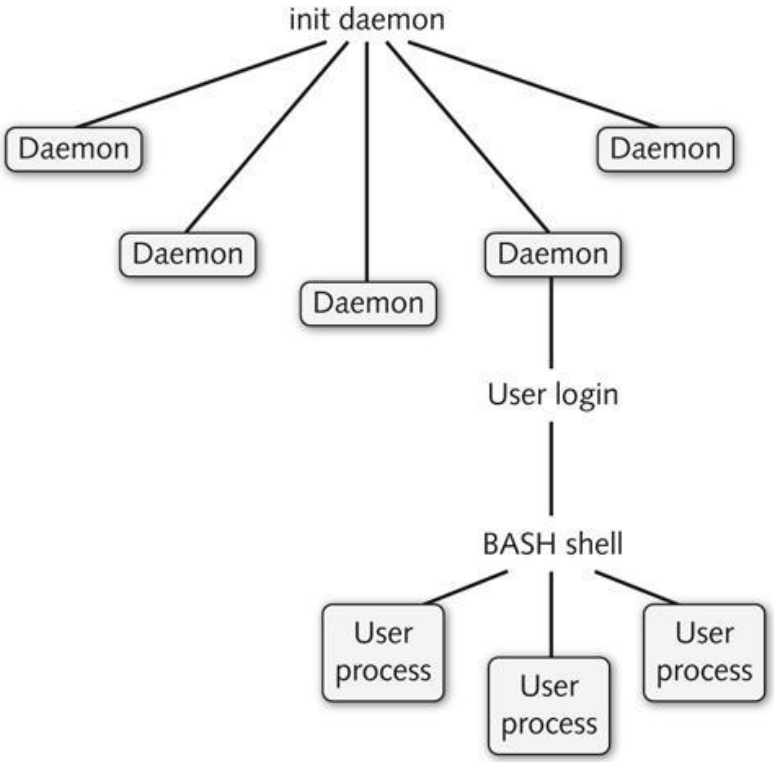


Figure 9-2: Process genealogy

Viewing Processes

- **ps**
  - View processes
  - Most versatile and common process viewing utility
    - Several other utilities
  - No arguments: lists all processes running in current shell
    - PID, terminal, command that started process, CPU time
  - ***–f*** (full) option
    - More complete information
      - User identifier (UID), PPID, start time, CPU utilization
  - ***-e*** option
    - Displays the entire list of processes across all terminals including daemons
  - ***-u*** option
    - Display list of processes that belong to certain user

Viewing Processes

- Process state (**ps *aux* *or* ps -l**)
  - Current processor state of process
  - Generally sleeping (**S**) or running (**R**)
  - This column is the **most valuable** to system administrators

- Zombie process
  - Process finished, but parent has not released child process's PID
  - Defunct process
  - Process state is **Z**

Process state codes:
https://web.archive.org/web/20150702131123/http://unixhelp.ed.ac.uk/CGI/man-cgi?ps

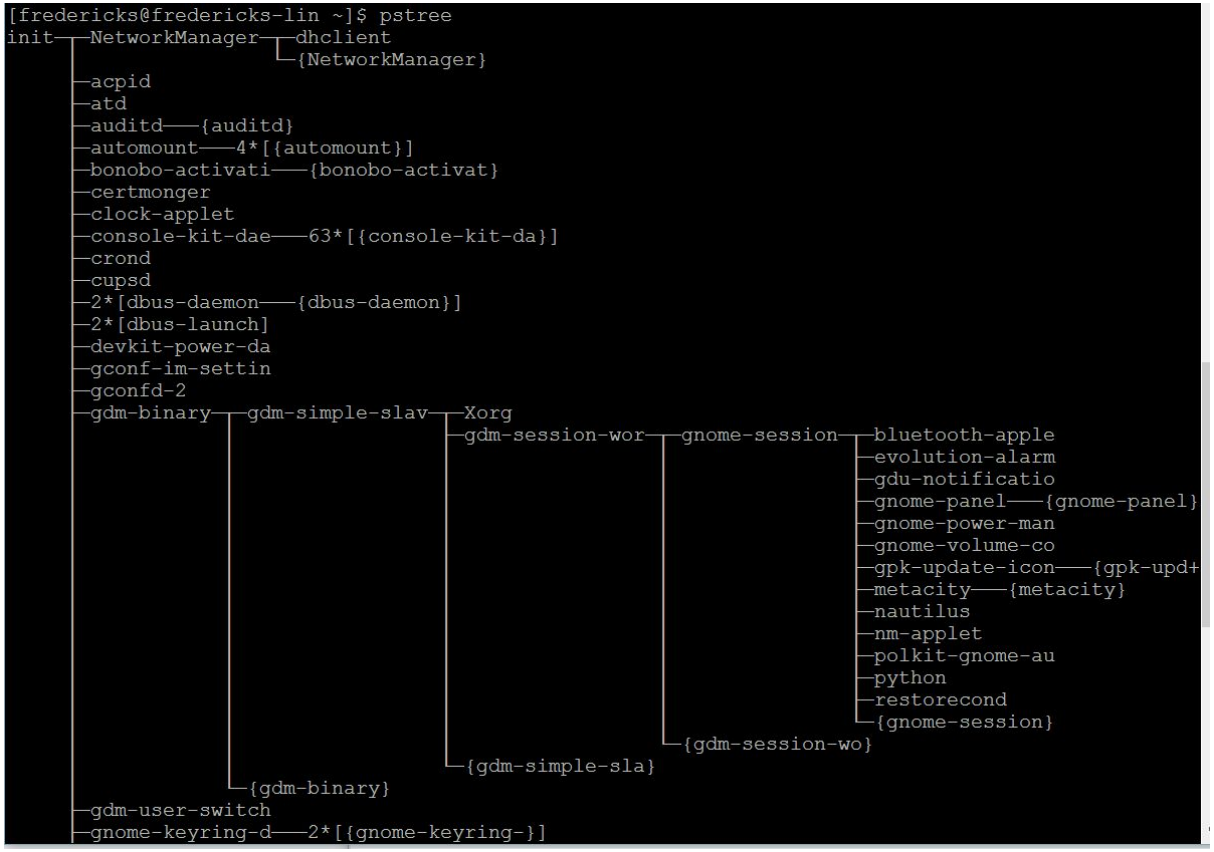
Viewing Processes

- Process priority (PRI)
  - Priority used by the kernel for the process
  - Higher value == lower priority
- Nice value (NI):
  - Can be used to affect the process priority ***indirectly***
  - Measured between -20 (a greater chance of a high priority) and 19 (a greater chance of a lower priority)
- Process size in memory (SZ) measured in kilobytes
- (This information is found with **ps –l**)

Viewing Processes

- Some options to the **ps** command are not prefixed by a dash
  - Referred to as Berkeley style options
- Two most common of these are:
  - The ***a*** option: lists all processes across terminals
  - The **x** option: lists processes that do not run on a terminal
- **pstree**
  - Show process lineage by tracing its PPIDs until the init daemon
  - -p: shows PIDs

pstree


[fredericks@fredericks-lin ~]$ pstree
init─┬─NetworkManager─┬─dhclient
│ └─{NetworkManager}
├─acpid
├─atd
├─auditd───{auditd}
├─automount───4*[{automount}]
├─bonobo-activati───{bonobo-activat}
├─certmonger
├─clock-applet
├─console-kit-dae───63*[{console-kit-da}]
├─crond
├─cupsd
├─2*[dbus-daemon───{dbus-daemon}]
├─2*[dbus-launch]
├─devkit-power-da
├─gconf-im-settin
├─gconfd-2
├─gdm-binary─┬─gdm-simple-slav─┬─Xorg
│ └─gdm-session-wor─┬─gnome-session─┬─bluetooth-apple
│ ├─evolution-alarm
│ ├─gdu-notificatio
│ ├─gnome-panel───{gnome-panel}
│ ├─gnome-power-man
│ ├─gnome-volume-co
│ ├─gpk-update-icon───{gpk-upd+
│ ├─metacity───{metacity}
│ ├─nautilus
│ ├─nm-applet
│ ├─polkit-gnome-au
│ ├─python
│ ├─restorecond
│ └─{gnome-session}
│ └─{gdm-session-wo}
│ └─{gdm-simple-sla}
│ └─{gdm-binary}
├─gdm-user-switch
├─gnome-keyring-d───2*[{gnome-keyring-}]

Viewing Processes

Option	Description
`-e`	Displays all processes running on terminals as well as processes that do not run on a terminal (daemons)
`-f`	Displays a full list of information about each process, including the UID, PID, PPID, CPU utilization, start time, terminal, processor time, and command name
`-l`	Displays a long list of information about each process, including the flag, state, UID, PID, PPID, CPU utilization, priority, nice value, address, size, WCHAN, terminal, and command name
`a`	Displays all processes running on terminals
`x`	Displays all processes that do not run on terminals

Table 9-1: Common options to the ps command

Viewing Processes

- **top**
  - Displays an interactive screen listing processes
  - Organized by processor time
  - Processes using most processor time are listed first
- Rogue process
  - Faulty process
  - Consumes excessive system resources
- **top** command can be used to change the priority of processes or to kill processes
  - Rogue processes can be killed immediately when identified

htop:

```
sudo yum install epel-release && sudo yum install htop
```

Killing Processes

- **kill**
  - Sends a kill signal to a process to terminate it
  - 64 types of kill signals
    - Each affects processes in different ways
  - ***-l*** option
    - Displays a list of kill signal names and associated numbers
  - To kill a process, give kill signal and PID
    - If no kill signal is given, the default kill signal, SIGTERM, is used

Killing Processes

Name	Number	Description
SIGHUP	1	Also known as the hang-up signal, it stops a process, then restarts it with the same PID. If you edit the configuration file used by a running daemon, that daemon might be sent a SIGHUP to restart the process; when the daemon starts again, it reads the new configuration file.
SIGINT	2	This signal sends an interrupt signal to a process. Although this signal is one of the weakest kill signals, it works most of the time. When you use the Ctrl+c key combination to kill a currently running process, a SIGINT is actually being sent to the process.
SIGQUIT	3	Also known as a core dump, the quit signal terminates a process by taking the process information in memory and saving it to a file called core on the hard disk in the current working directory. You can use the Ctrl+\ key combination to send a SIGQUIT to a process that is currently running.
SIGTERM	15	The software termination signal is the most common kill signal used by programs to kill other processes. It is the default kill signal used by the `kill` command.
SIGKILL	9	Also known as the absolute kill signal, it forces the Linux kernel to stop executing the process by sending the process's resources to a special device file called /dev/null.

Table 9-2: Common administrative kill signals

Killing Processes

- Trap: the ability of a process to ignore a kill signal
  - The SIGKILL signal cannot be trapped by any process
    - Use only as last resort because it prevents a process from closing temporary files and resources properly
- When a parent process receives a kill signal
  - Terminates all child processes before terminating self
  - To kill several related processes send signal to parent process
  - Kill parent process in order to kill zombie processes

Killing Processes

- **killall**
  - Kills multiple processes of the same name in one command
  - Takes kill signal number as an option
  - Uses process name instead of PID
  - If no kill signal given, the default kill signal, SIGTERM, is used
  - E.g., kill all instances of script if it is parallelized

- Can use **top** command to kill processes
  - While in the **top** utility, press the k key and supply the appropriate PIS and kill signal when prompted

Process Execution

- Forking
  - Act of creating new BASH shell or subshell
  - Carried out by the fork function in the BASH shell
  - Subshell executes program or shell script using **exec** function
  - Original shell uses its wait function to wait for the new BASH shell (subshell) to complete
  - When done, subshell kills itself
    - Control returns to the original shell

Process Execution

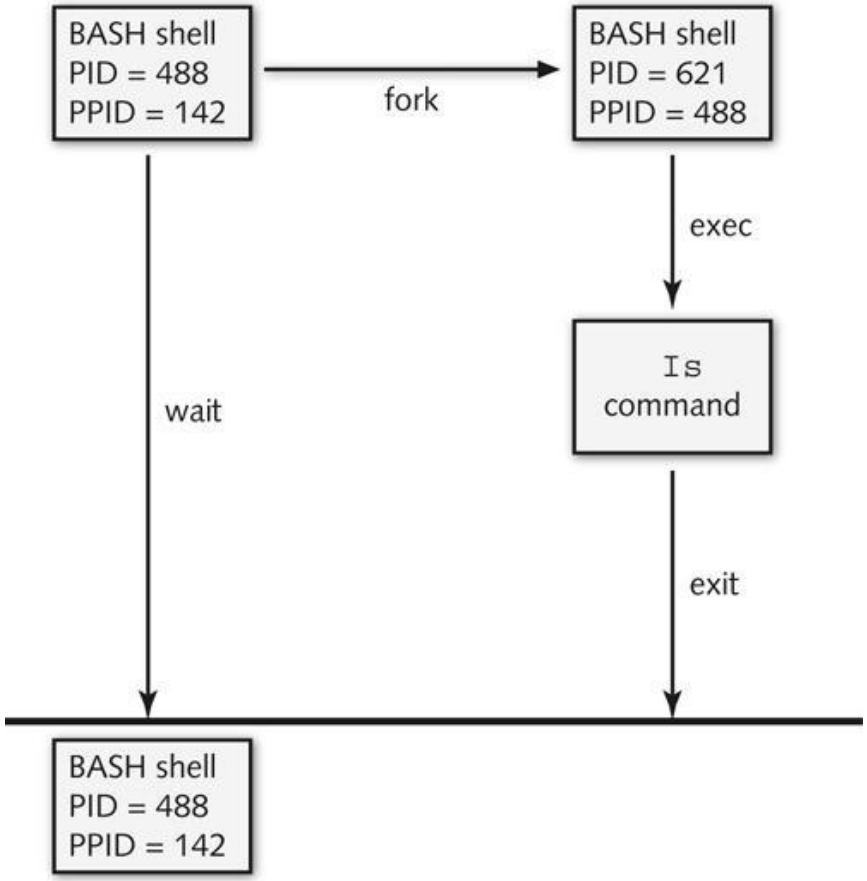


Figure 9-3: Process forking

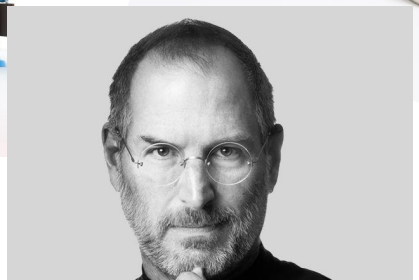
Running Processes in the Background

- Foreground processes
  - BASH shell must wait for termination to display prompt and accept new commands
- Background processes
  - BASH shell does not wait for termination
  - Append an ampersand (&) character to the command
  - Upon execution, user receives BASH shell prompt immediately
    - Can enter commands again

- ./long_running_script → wait until completion
- ./long_running_script & → runs in background

Running Processes in the Background

- **jobs**
  - Lists background job IDs for processes running in current shell
- To terminate a background process:
  - Send a kill signal to the PID
  - You can also send a kill signal to background job ID
    - Prefix job ID with a percent (%) character



Running Processes in the Background

- **foreground (fg)**
  - Moves a background process to the foreground
- **background (bg)**
  - Send an existing process to the background
  - Provide background job ID as argument

Process Priorities

- Time slice
  - Amount of time a process is given on a CPU
  - More time slices mean more execution time on CPU
    - Executes faster
  - Usually measured in milliseconds

Process Priorities

- PRI(ority) dictates number of time slices a process gets
  - Low PRI is likely to get more time slices than high PRI
  - Cannot change PRI value directly
  - Set the nice value (NI) to indirectly affect priority
    - A negative NI value, more time slices; a positive NI value, less time slices
- Processes start with NI of 0 by default
- **nice**
  - Change a process's priority as it starts

Process Priorities

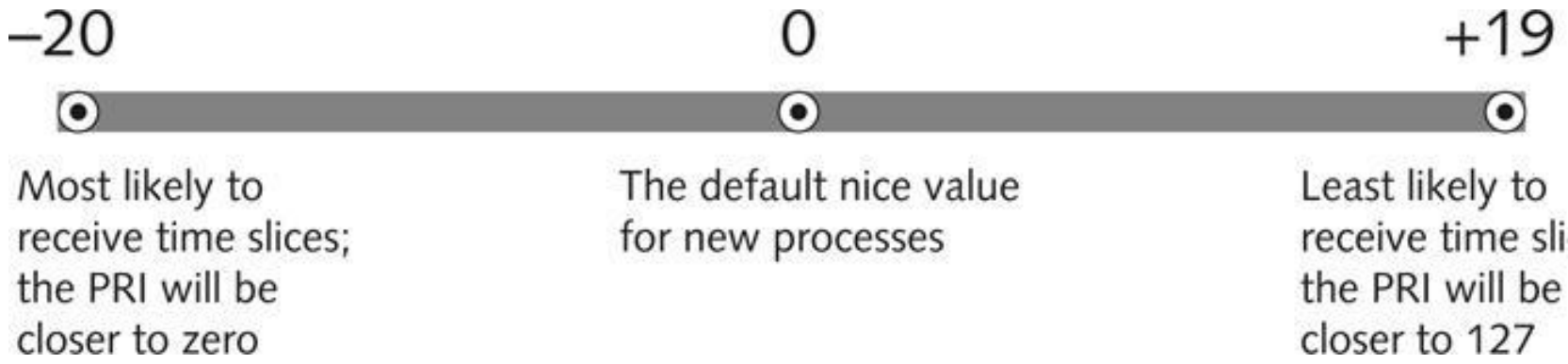


Figure 9-4: The nice value scale

Process Priorities

- **renice**
  - Alter NI of a process after it has been started
  - Only root user may change NI to a negative value
  - -u option: change the NI for all processes owned by the specified user or group
- May also change NI of running processes using **top**
  - Press the **r** key, then supply the PID and the nice value when prompted

Scheduling Commands

- Scheduling system maintenance commands to run during nonworking hours is good practice
- To schedule commands to execute in the future:
  - **at** daemon (atd)
    - System daemon that executes tasks at a future time
    - E.g., email user when task is done
  - **cron** daemon (crond)
    - System daemon that executes tasks repetitively in the future
    - cron job
    - E.g., backup system offsite weekly

Scheduling Commands with atd

- **at**
  - Schedule commands and tasks to run at a preset time
  - Specify the time as an argument
- After being invoked, the at command displays an **at>** prompt
  - Allows you to type commands to be executed
  - One per line
- After commands have been entered
  - Use the **Ctrl+d** key combination to schedule the commands

Scheduling Commands with atd

Command	Description
`at 10:15pm`	Schedules commands to run at 10:15 PM on the current date
`at 10:15pm July 15`	Schedules commands to run at 10:15 PM on July 15
`at midnight`	Schedules commands to run at midnight on the current date
`at noon July 15`	Schedules commands to run at noon on July 15
`at teatime`	Schedules commands to run at 4:00 PM on the current date
`at tomorrow`	Schedules commands to run the next day
`at now + 5 minutes`	Schedules commands to run in five minutes
`at now + 10 hours`	Schedules commands to run in 10 hours
`at now + 4 days`	Schedules commands to run in four days
`at now + 2 weeks`	Schedules commands to run in two weeks
`at now` `at batch`	Schedules commands to run immediately

`at 9:00am 01/03/2016`
`at 9:00am 01032016`
`at 9:00am 03.01.2016`
	Schedules commands to run at 9:00 AM on January 3, 2016

Table 9-3: Common at commands

Scheduling Commands with atd

- If the /etc/at.allow and /etc/at.deny files do not exist
  - Only the root user is allowed to schedule tasks using the at daemon
- To give this ability to other users
  - Create an /etc/at.allow file and add the names of users allowed to use the at daemon, one per line
  - Use the /etc/at.deny file to deny certain users access to the at daemon
- If both files exist, the system checks the at.allow file and does not process the at.deny file
  - at.deny exists in your systems following install

Scheduling Commands with cron

- Schedule repetitive tasks
  - Uses configuration files called cron tables to specify when a command should be executed
- Cron tables include:
  - Six fields separated by space or tab characters
    - First five specify times to run the command
    - Sixth absolute pathname to command to be executed

Scheduling Commands with cron

- Two types of cron tables are used by the cron daemon
  - User cron tables
    - Tasks scheduled by individual users
  - System cron tables
    - Contains system tasks
- /var/spool/cron
  - /var/spool/cron/crontabs: on Ubuntu systems
  - Individual crons for user
- /etc/crontab file
  - Contains system cron tables
- /etc/cron.d directory
  - Contains system cron tables

Scheduling Commands with cron

1 2 3 4 5 command

1 = Minute past the hour (0–59)
2 = Hour (0–23)
3 = Day of month (1–31)
4 = Month of year (1–12)
5 = Day of week
0 = Sun (or 7 = Sun)
1 = Mon
2 = Tues
3 = Wed
4 = Thurs
5 = Fri
6 = Sat

Figure 9-5: User cron table format

Scheduling Commands with cron

Run /usr/bin/some_script at 4:01am, 5:01am, 4:31am, 5:31am on the 1st through 15th every January and June

Run continuously (1st through 15th)

```
01,31 04,05 1-15 1,6 * /usr/bin/some_script
```

1 2 3 4 5 command

1 = Minute past the hour (0–59)
2 = Hour (0–23)
3 = Day of month (1–31)
4 = Month of year (1–12)
5 = Day of week
0 = Sun (or 7 = Sun)
1 = Mon
2 = Tues
3 = Wed
4 = Thurs
5 = Fri
6 = Sat

Run multiple instances (X:01, X:31)

All entries (each day of the week)

User Cron Tables

- /etc/cron.allow:
  - Lists users allowed to use the cron daemon
- /etc/cron.deny:
  - Lists users not allowed to use the cron daemon
- If both files exist, only the /etc/cron.allow file is processed

User Cron Tables

- **crontab** command: use to create or edit user cron tables
  - –e option: edit cron tables in vi editor
  - –l option: list a user cron table
  - –r option: remove cron table and all scheduled jobs
  - -u option: used by root user to edit, list, or remove a specified user's cron table

*/1 * * * * ls

→ send to user mail every 1 minute

*/1 * * * * ls > /home/username/logfile.log

→ dump output to log file every 1 minute

The System Cron Table

- System maintenance, backups, and CPU-intensive tasks
  - Schedule for non-business hours or odd times
  - System admin tasks…

- Typically scheduled by the cron daemon
  - From entries in system cron table (/etc/crontab)
    - These entries can only be edited by the root user

The System Cron Table

- Initial section of cron table specifies execution environment
  - Remainder of the file contains comments that identify the format of a cron table entry

- Cron tables located in the /etc/cron.d directory are run by the system as a specified user

The System Cron Table

- Initial section of cron table specifies execution environment
  - Remainder of the file contains comments that identify the format of a cron table entry

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
HOME=/

# For details see man 4 crontabs

# Example of job definition:
# .---------------- minute (0 - 59)
# |  .------------- hour (0 - 23)
# |  |  .---------- day of month (1 - 31)
# |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
# |  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,f
ri,sat
# |  |  |  |  |
# *  *  *  *  * user-name command to be executed
```

The System Cron Table

- System tasks can be performed hourly, daily, weekly, or monthly
  - For this, you don't need to create a system cron table

- Place script in:
  - /etc/cron.hourly/
  - /etc/cron.daily/
  - /etc/cron.weekly
  - etc/cron.monthly/

An enjoyable in-class assignment

What are **two** things you would schedule?


'MOVIN'
RIGHT
ALONG
muppetstuff

System Startup

- Summarize the major steps necessary to boot a Linux system
- Detail the configuration of common Linux bootloaders
- Explain the UNIX SysV and Systemd system initialization processes
- Start, stop, and restart daemons
- Configure the system to start and stop daemons upon entering certain runlevels and targets

The Boot Process

- POST (Power On Self Test)
  - Series of tests run when computer initializes
  - Ensures functionality of hardware
- Bootloader
  - Program used to load an OS
  - Locates and executes the kernel of the OS
  - MBR might contain pointer to a partition containing a bootloader on the first sector
- Active partition
  - Partition pointed to by MBR
  - One per HDD

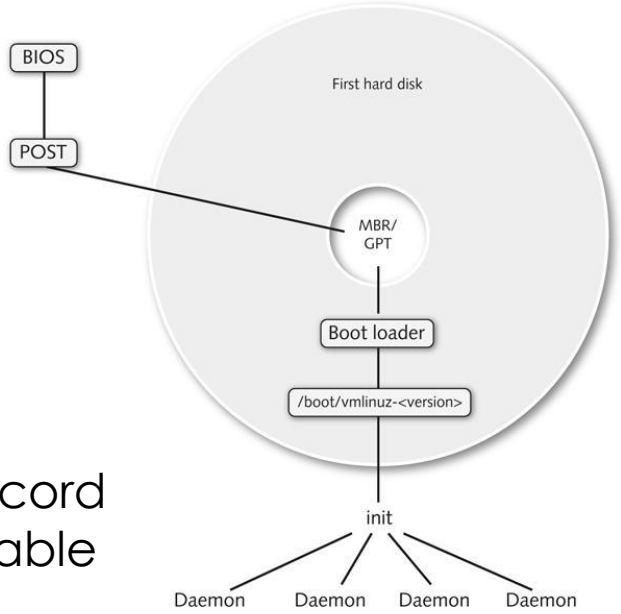
The Boot Process (Terms)

- /boot
  - Directory containing kernel and boot-related files
- vmlinuz-<kernel version>
  - Linux kernel file

- Daemon
  - System process that performs useful tasks
    - e.g., printing, scheduling, OS maintenance

- Init (initialize) daemon
  - First process started by Linux kernel
  - Loads all other daemons
  - Brings system to usable state

The Boot Process



MBR = Master Boot Record
GPT = GUID Partition Table

Figure 8-1: The boot process

Bootloaders

- Primary function of bootloader
  - Load Linux kernel into memory
- Other functions
  - Passing information to the kernel during startup
  - Booting another OS
    - Dual booting
- Most common bootloaders:
  - GRand Unified Bootloader (GRUB)/GRUB2
  - Linux Loader (LILO)

LILO

- Linux Loader
- Traditional Linux bootloader
  - No longer used by most modern Linux distributions
  - Often found on other distributions that require a smaller bootloader than GRUB
- Typically located on MBR/GPT
- Lilo boot: prompt appears following BIOS POST
  - Allows choice of OS to load at startup
- To configure, edit /etc/lilo.conf file

LILO


LILO Boot Menu
Linux
LinuxOLD
windows
Hit any key to cancel timeout --:--
Use ←↑↓→ arrow keys to make selection
Enter choice & options, hit CR to boot
boot:

LILO

Keyword	Description
image=	Specifies the absolute pathname to the Linux kernel.
root=	Specifies the device and the partition that contains the Linux root filesystem.
prompt	Displays a LILO boot prompt provided there is no message= keyword specified.
message=	Specifies the absolute pathname to the file that contains a graphical LILO screen that can be used instead of a prompt. Y; you can press Ctrl+x at this graphical screen to switch to the LILO boot: prompt.
timeout=	Specifies the number of 1/10th seconds to wait for user input before loading the default operating system kernel.
default=	Specifies the label for the default operating system kernel.
label=	Specifies the friendly name used to identify an operating system kernel within boot loader screens.
boot=	Specifies where LILO should be installed. If the device specified is a partition on a hard disk, LILO is installed at the beginning of the partition. If the device specified is a disk, LILO is installed to the MBR on that device.
linear	Specifies that LILO uses linear sector addressing.
read-only	Initially mounts the Linux root filesystem as read-only to reduce any errors with running fsck during system startup.
initrd=	Specifies the pathname to a ramdisk image used to load modules into memory needed for the Linux kernel at boot time.
password=	Specifies a password required to boot the Linux kernel.
append=	Specifies parameters that are passed to the Linux kernel when loaded.
map=	Specifies the file that contains the exact location of the Linux kernel on the hard disk.
install=	Specifies the file that contains the physical layout of the disk drive.
lba32	Specifies large block addressing (32-bit) for hard drives that have more than 1024 cylinders.

Table 8-1: Common /etc/lilo.conf keywords

LILO Example (lilo.conf)

- (Scientific Linux 7 uses GRUB)

```
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
prompt
timeout=50
message=/boot/message
lba32
default=linux
image=/boot/vmlinuz-2.4.0-0.43.6
label=linux
initrd=/boot/initrd-2.4.0-0.43.6.img
read-only
root=/dev/hda5
other=/dev/hda1
label=dos
```

LILO

- Must reinstall LILO if /etc/lilo.conf file altered
- **lilo** command: Reinstalls LILO
  - ***-u*** option: Uninstall LILO

LILO

Error Message	Description
L	The first part of the LILO boot loader failed to load, usually as a result of incorrect hard disk parameters. Simply rebooting the machine sometimes fixes this problem. However, you might also need to add the word "linear" to /etc/lilo.conf.
LI	The second part of the LILO boot loader failed to load or the /boot/boot.b file is missing. Adding the word "linear" to /etc/lilo.conf might fix the problem.
LIL LIL- LIL?	LILO has loaded properly but cannot find certain files required to operate, such as the /boot/map and /boot/boot.b files. Adding the word "linear" to /etc/lilo.conf might fix the problem.

Table 8-2: LILO error messages

GRUB

- Grand Unified Bootloader
- Originally created to replace LILO
- Stage1
  - First major part of GRUB
  - Typically resides on MBR/GPT
  - Remaining parts of the bootloader (Stage1.5 and Stage2) reside in the /boot/grub directory
- Stage1.5
  - Loads filesystem support and Stage2
- Stage2
  - Performs the actual bootloader functions
  - Displays graphical bootloader screen

GRUB

Figure 8-2: GRUB bootloader screen

GRUB

- To configure, edit the /boot/grub/grub.cfg file
  - Read directly by Stage2 bootloader
  - HDDs and partitions are identified by numbers
    - Format: (hd<drive#>,<partition#>)

- GRUB normally allows manipulation of bootloader during system startup
  - To prevent this, password protect GRUB modifications during boot time

GRUB

```
# grub.conf generated by anaconda
#
# Note that you do not have to rerun grub after making changes to this file
# NOTICE:  You have a /boot partition.  This means that
#          all kernel and initrd paths are relative to /boot/, eg.
#          root (hd0,0)
#          kernel /vmlinuz-version ro root=/dev/mapper/vg_frederickslin-lv_root
#          initrd /initrd-[generic-]version.img
#boot=/dev/sda
default=0
timeout=5
splashimage=(hd0,0)/grub/splash.xpm.gz
hiddenmenu
title Scientific Linux 6 (2.6.32-573.el6.x86_64)
        root (hd0,0)
        kernel /vmlinuz-2.6.32-573.el6.x86_64 ro root=/dev/mapper/vg_fredericksl
in-lv_root rd_NO_LUKS LANG=en_US.UTF-8 rd_LVM_LV=vg_frederickslin/lv_root rd_NO_
MD SYSFONT=latarcyrheb-sun16 crashkernel=128M rd_LVM_LV=vg_frederickslin/lv_swap
  KEYBOARDTYPE=pc KEYTABLE=us rd_NO_DM rhgb quiet
        initrd /initramfs-2.6.32-573.el6.x86_64.img
```

f file

during

- To prevent this, password protect GRUB modifications during boot time

GRUB

- If you press any key during first five seconds after the BIOS POST, you will get a graphical GRUB boot menu screen
  - Allows you to manipulate the boot process
  - grub> prompt to enter commands
    - Help screen provides list of all available commands

- **grub-install**
  - Installs GRUB bootloader
  - Typically for reinstallation when GRUB becomes damaged

GRUB

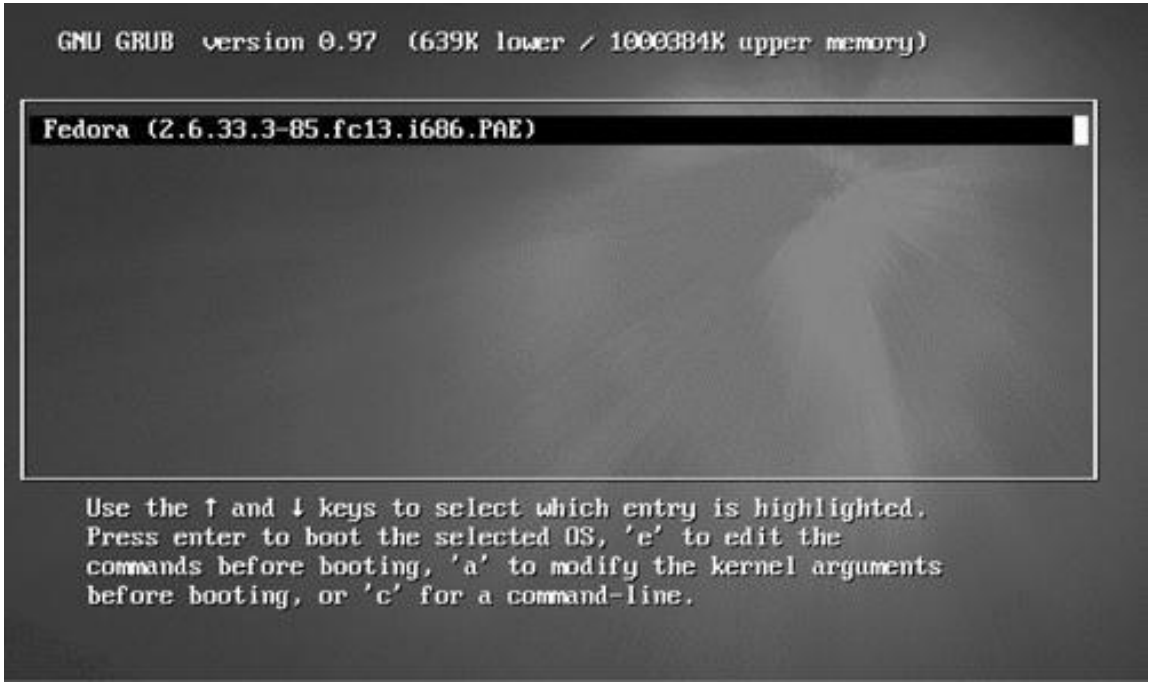


Figure 8-3: The GRUB boot menu for a Fedora system

GRUB2

- GRUB2
  - Most common bootloader used on modern Linux systems
- Similar structure to GRUB
  - Stage2 loads a terminal-friendly bootloader screen
- Main configuration file for GRUB2 is /boot/grub/grub.cfg (or /boot/grub2/grub.cfg)
- When a new device driver needs to be loaded by the bootloader
  - The package often adds a file to /etc/default/grub.d

GRUB2

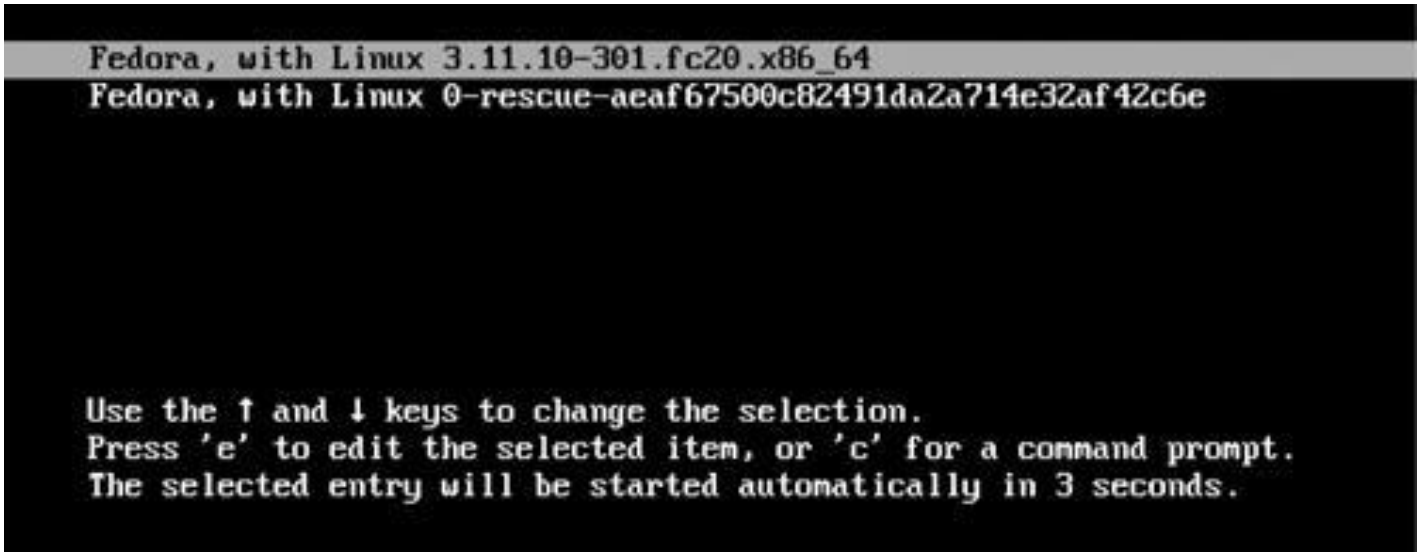


Figure 8-4: The GRUB2 boot screen on a Fedora 20 system

GRUB2

- To configure or edit:
  - Add or modify existing lines within the /etc/default/grub file
- After modifying the file or adding scripts to the /etc/grub.d directory
  - Run the **grub2-mkconfig** command to rebuild the /boot/grub/grub.cfg file
- Use the **grub2-install** command
  - If the GRUB2 bootloader becomes damaged

Which Version of GRUB?

- We use GRUB 1 (0.97)
- **$ grub2-install *–version***
  - grub-install (GNU GRUB 0.97)
- (Ubuntu uses GRUB2)

Linux Initialization

- The kernel resumes control after Linux is loaded
  - init daemon (first process) executed
  - Performs a system initialization process
    - Brings the system into a usable state
- Recent Linux distributions have adopted the Systemd system initialization processes
  - Older Linux systems used a UNIX standard called SysV
    - Upstart is a variant of SysV
- Systemd is completely compatible with SysV
  - Implements new features for management

Which Version?

- RPM package manager: check what runs /sbin/init
  - fedora:~$ **rpm** -qf /sbin/init
    - systemd-216-24.fc21.x86_64
  - scientific:~$ **rpm** -qf /sbin/init
    - **systemd-219-57.el7_5.3.x86_64**
  - opensuse:~$ **rpm** -qf /sbin/init
    - systemd-sysvinit-44-10.1.1.i586
- dpkg (Debian systems)
  - **dpkg** –S /sbin/init
    - upstart: /sbin/init

Working with the UNIX SysV System Initialization Process

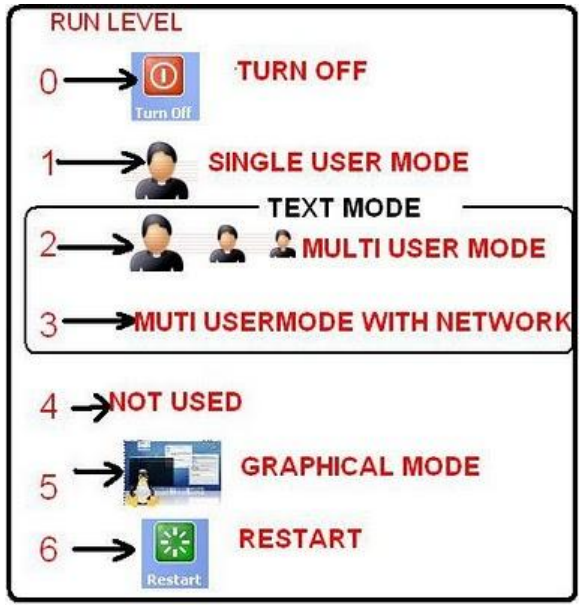
- Two SysV system initialization processes
  - Traditional SysV
  - Upstart
    - Default in RHEL6
- In both systems
  - The init daemon runs a series of scripts to start other daemons to provide system services
- The init daemon is responsible for starting and stopping daemons after system initialization

Runlevels

- Runlevel
  - Defines the number and type of daemons loaded into memory and executed by the kernel
- The init daemon responsible for changing runlevels
  - Often called initstates
- Seven standard runlevels
- **runlevel**
  - Displays current and most recent runlevel
- **init**
  - Changes the runlevel on a system
  - **telinit** command: Alias to **init** command

Runlevels

- Describe state of system
- Level 0
  - System completely down
- Level 1 (S)
  - Single-user mode
- Level 2-5
  - Multi-user levels
  - 3 is common for CLI login
  - 5 is common for everyday use
- Level 6
  - Reboot level



Runlevels

Runlevel	Common Name	Description
0	Halt	A system that has no daemons active in memory and is ready to be powered off.
1 s S single	Single User Mode	A system that has only enough daemons to allow one user (the root user) to log in and perform system maintenance tasks.
2	Multiuser Mode	A system that has most daemons running and allows multiple users the ability to log in and use system services. Most common network services other than specialized network services are available in this runlevel as well.
3	Extended Multiuser Mode	A system that has the same abilities as Multiuser Mode, yet with all extra networking services started (e.g., SNMP, NFS).
4	Not used	Not normally used, but can be customized to suit your needs.
5	Graphical Mode	A system that has the same abilities as Extended Multiuser Mode, yet with a graphical login program. On systems that use the GNOME desktop, this program is called the **GNOME Display Manager (gdm)** and is started on tty1 or tty7 to allow for graphical logins.
6	Reboot	A special runlevel used to reboot the system.

Table 8-3: Linux runlevels

The /etc/inittab File

- Unless otherwise specified…
  - init daemon enters the default runlevel indicated in the /etc/inittab file
  - Syntax: `id:5:initdefault:`
- Contains a single uncommented line that configures the default runlevel
- Switch runlevels via command line?
  - sudo init 3 → goes to RL3

The /etc/inittab File

systemctl set-default <runlevel>.target

- ~~Unless otherwise specified…~~
  - ~~init daemon enters the default runlevel indicated in the /etc/inittab file~~
  - ~~Syntax: `id:5:initdefault:`~~
- ~~Contains a single uncommented line that configures the default runlevel~~

- ~~Switch runlevels via command line?~~
  - ~~sudo init 3 → goes to RL3~~

Runtime Configuration Scripts

- Runtime configuration (rc) scripts
  - Prepare the system
  - Start daemons
  - Bring the system to a usable state
- init daemon executes script for default runlevel (5)
  - Executes all files that start with S (start) in the /etc/rc.d/rc5.d directory
  - When system is shut down (0) / restarted (6), scripts that start with (K) are run (kill process)

Runtime Configuration Scripts

- Runtime configuration (rc) scripts

```
[fredericks@fredericks-lin ~]$ sudo ls /etc/rc.d/rc5.d/
[sudo] password for fredericks:
K01smartd          K76ypbind          S11auditd          S26haldaemon
K02oddjobd         K84wpa_supplicant  S11portreserve     S26udev-post
K05wdaemon         K87restorecond     S12rsyslog         S28autofs
K10psacct          K88sssd            S13cpuspeed        S50bluetooth
K10saslauthd       K89netconsole      S13irqbalance      S50kdump
K15htcacheclean    K89rdisc           S13rpcbind         S55sshd
K15httpd           K92pppoe-server    S15mdmonitor       S70spice-vdagentd
K50dnsmasq         K95firstboot       S22messagebus      S80postfix
K60nfs             K95rdma            S23NetworkManager  S90crond
K61nfs-rdma        K99rngd            S24nfslock         S95atd
K69rpcsvcgssd      S01sysstat         S24rpcgssd         S99certmonger
K73winbind         S02lvm2-monitor    S25blk-availability S99local
K74ntpd            S08ip6tables       S25cups
K75ntpdate         S08iptables        S25netfs
K75quota_nld       S10network         S26acpid
```

5)

he

Runtime Configuration Scripts

- When user specifies runlevel1…
  - init daemon runs default script
  - Executes files in the /etc/rc.d/rc1.d directory
- Each file in an /etc/rc.d directory is a symbolic link to a script that can be used to start or stop a daemon
  - Depending on whether the symbolic link filename started with an S (start) or K (kill/stop)

Runtime Configuration Scripts

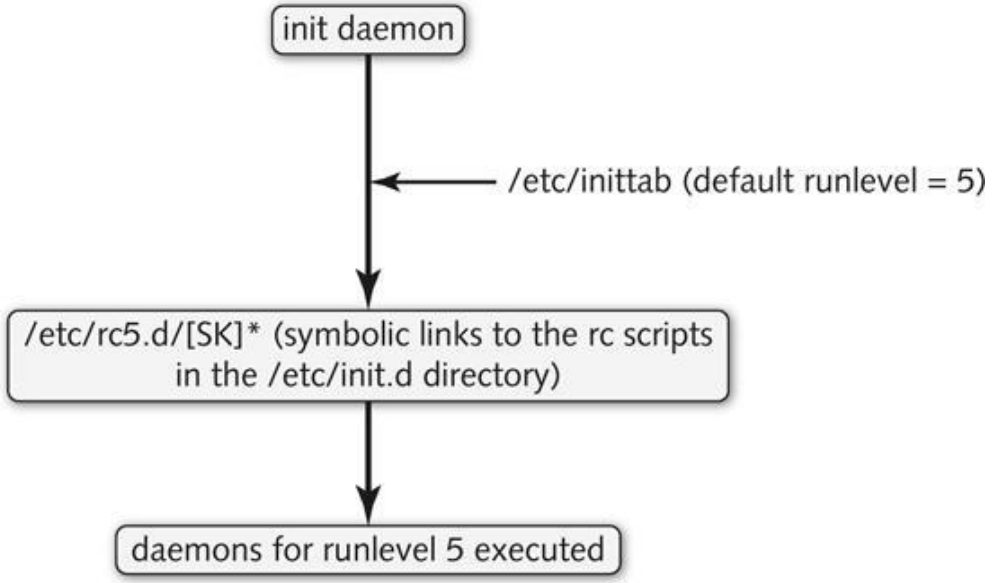


Figure 8-5: A traditional UNIX SysV system initialization process

Runtime Configuration Scripts

- On Linux systems that use the upstart init system
  - The /etc/rc.d directories are **not** used
- The init daemon identifies the default runlevel in the /etc/inittab file
  - Directly executes the rc scripts within the /etc/init.d directory
- Each daemon has a separate configuration file within the /etc/init directory
  - Identify runlevels it should be started or stopped in
  - E.g., etc/init/splash-manager.conf

Runtime Configuration Scripts

- On Linux systems that use the upstart init system

```
[fredericks@fredericks-lin ~]$ ls /etc/init.d
acpid              haldaemon      netconsole      psacct        single
atd                halt           netfs           quota_nld     smartd
auditd             htcacheclean   network         rdisc         spice-vdagentd
autofs             httpd          NetworkManager  rdma          sshd
blk-availability   ip6tables      nfs             restorecond   sssd
bluetooth          iptables       nfslock         rngd          sysstat
certmonger         irqbalance     nfs-rdma        rpcbind       udev-post
cpuspeed           kdump          ntpd            rpcgssd       wdaemon
crond              killall        ntpdate         rpcidmapd     winbind
cups               lvm2-lvmetad   oddjobd         rpcsvcgssd    wpa_supplicant
dnsmasq            lvm2-monitor   portreserve     rsyslog       ypbind
firstboot          mdmonitor      postfix         sandbox
functions          messagebus     pppoe-server    saslauthd
```

the /etc/inittab

directory

- Each daemon has a separate configuration file within the /etc/init directory
  - Uses standard wildcard notation to identify runlevels it should be started or stopped in

Runtime Configuration Scripts

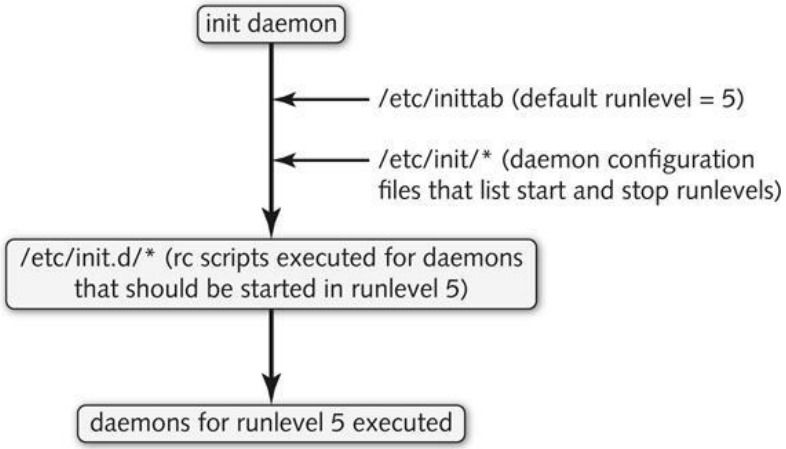


Figure 8-6: An upstart system initialization process

Starting and Stopping Daemons Manually

- Most daemons accept arguments **start**, **stop**, **restart**
  - Can be used to manipulate daemons after system startup
- E.g., to restart the cron daemon:
  - **/etc/init.d/cron restart**
- **service**
  - Start, stop, or restart daemons within /etc/rc.d/init.d directory
- Upstart system also provides the **stop**, **start**, and **restart** commands

Configuring Daemons to Start in a Runlevel

- To configure a daemon to start or stop in a particular runlevel:
  - Create or modify symbolic links within /etc/rc[runlevel].d directories
    - *Not for Upstart…*
- **chkconfig**
  - View and modify daemons that are started in each runlevel
    - *Use for Upstart…*
- The **chkconfig** command is not available in Ubuntu Server 14.04
  - Use the **update-rc.d** command to configure files within /etc/rc[runlevel].d directories

(For us, it's: `$ systemctl list-unit-files`)

Configuring Daemons to Start in a

```
[fredericks@fredericks-lin ~]$ chkconfig
NetworkManager  0:off   1:off   2:on    3:on    4:on    5:on    6:off
acpid           0:off   1:off   2:on    3:on    4:on    5:on    6:off
atd             0:off   1:off   2:off   3:on    4:on    5:on    6:off
auditd          0:off   1:off   2:on    3:on    4:on    5:on    6:off
autofs          0:off   1:off   2:off   3:on    4:on    5:on    6:off
blk-availability        0:off   1:on    2:on    3:on    4:on    5:on    6:off
bluetooth       0:off   1:off   2:off   3:on    4:on    5:on    6:off
certmonger      0:off   1:off   2:off   3:on    4:on    5:on    6:off
cpuspeed        0:off   1:on    2:on    3:on    4:on    5:on    6:off
crond           0:off   1:off   2:on    3:on    4:on    5:on    6:off
cups            0:off   1:off   2:on    3:on    4:on    5:on    6:off
dnsmasq         0:off   1:off   2:off   3:off   4:off   5:off   6:off
firstboot       0:off   1:off   2:off   3:off   4:off   5:off   6:off
haldaemon       0:off   1:off   2:off   3:on    4:on    5:on    6:off
htcacheclean    0:off   1:off   2:off   3:off   4:off   5:off   6:off
httpd           0:off   1:off   2:off   3:off   4:off   5:off   6:off
ip6tables       0:off   1:off   2:on    3:on    4:on    5:on    6:off
iptables        0:off   1:off   2:on    3:on    4:on    5:on    6:off
irqbalance      0:off   1:off   2:off   3:on    4:on    5:on    6:off
kdump           0:off   1:off   2:on    3:on    4:on    5:on    6:off
lvm2-monitor    0:off   1:on    2:on    3:on    4:on    5:on    6:off
mdmonitor       0:off   1:off   2:on    3:on    4:on    5:on    6:off
messagebus      0:off   1:off   2:on    3:on    4:on    5:on    6:off
netconsole      0:off   1:off   2:off   3:off   4:off   5:off   6:off
netfs           0:off   1:off   2:off   3:on    4:on    5:on    6:off
network         0:off   1:off   2:on    3:on    4:on    5:on    6:off
```

ular runlevel:

el].d directories

ns that are

ntu Server

within

Working with the Systemd System Initialization Process

- Systemd is similar to SysV
  - Can also be used to start, stop, and configure many other OS components
  - **DEFAULT IN SCIENTIFIC LINUX 7**
- Each OS component is called a unit
- Daemons are called service units
- Runlevels are called target units (or targets)

- Typical OSs: ArchLinux, Debian, Fedora, openSUSE, Ubuntu…
  - https://en.wikipedia.org/wiki/Systemd#Adoption_and_reception

Working with the Systemd System Initialization Process

- Each target maps to a UNIX SysV runlevel:
  - poweroff.target = Runlevel 0
  - rescue.target = Runlevel1 (Single User Mode)
  - multi-user.target = Runlevel 2, 3, and 4
  - graphical.target = Runlevel 5
  - reboot.target = Runlevel 6
- Default target on a system with a GUI installed is the graphical.target
- To configure a different target:
  - Check what is available: # `systemctl list-units --type=target`
  - Change to GUI: `# systemctl set default graphical.target`

Working with the Systemd System Initialization Process

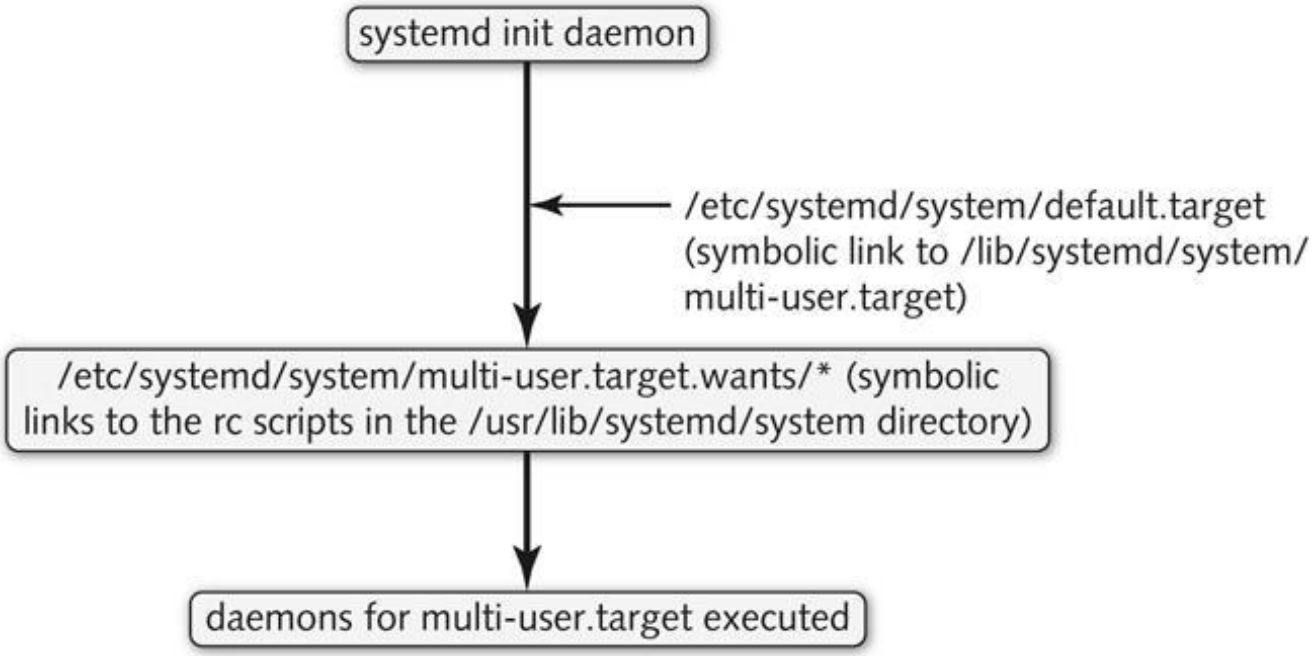


Figure 8-7: A Systemd system initialization process

Working with the Systemd System Initialization Process

- **systemctl**
  - Used to start and stop daemons, as well as configure them to automatically start during system initialization
- Syntax:
  - **systemctl restart crond.service**
- **systemctl** arguments
  - **status**
    - Detailed information about a daemon
  - **enable**
    - Configure a daemon to start in the default target
  - **isolate**:
    - Change between targets
    - Switch to graphics: `sudo systemctl isolate graphical.target`

The X Windows System: Linux GUI Components

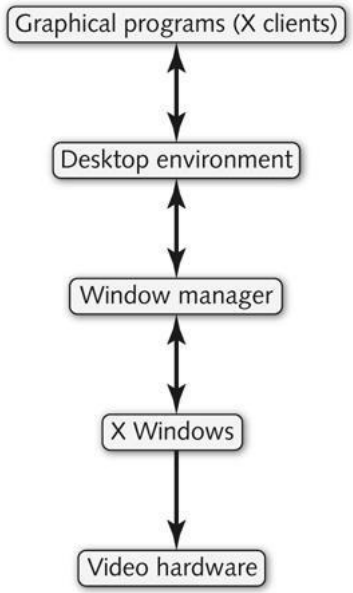


Figure 8-8: Components of the Linux GUI

X Windows

- X Windows
  - Core component of Linux GUI
  - Draw graphical images in windows that are displayed on terminal screen
  - Sometimes referred to as X server
- X client
  - Programs that tell X Windows how to draw the graphics and display the results
  - Need not run on same computer as X Windows
- XFree86
  - OSS version of X Windows
  - Originally intended for Intel x86 platform

Windows Managers and Desktop Environments

- Window manager
  - Modifies look and feel of X Windows
  - Fluxbox, Xfwm are common

- Desktop environment
  - Standard set of GUI tools
  - Works with a window manager to provide standard GUI environment
  - Provides toolkits that speed up process of creating new software
  - KDE, GNOME, Unity are the most common

SIGH:

```
sudo yum install epel-release -y
```

```
sudo yum groupinstall "X Window system" -y
```

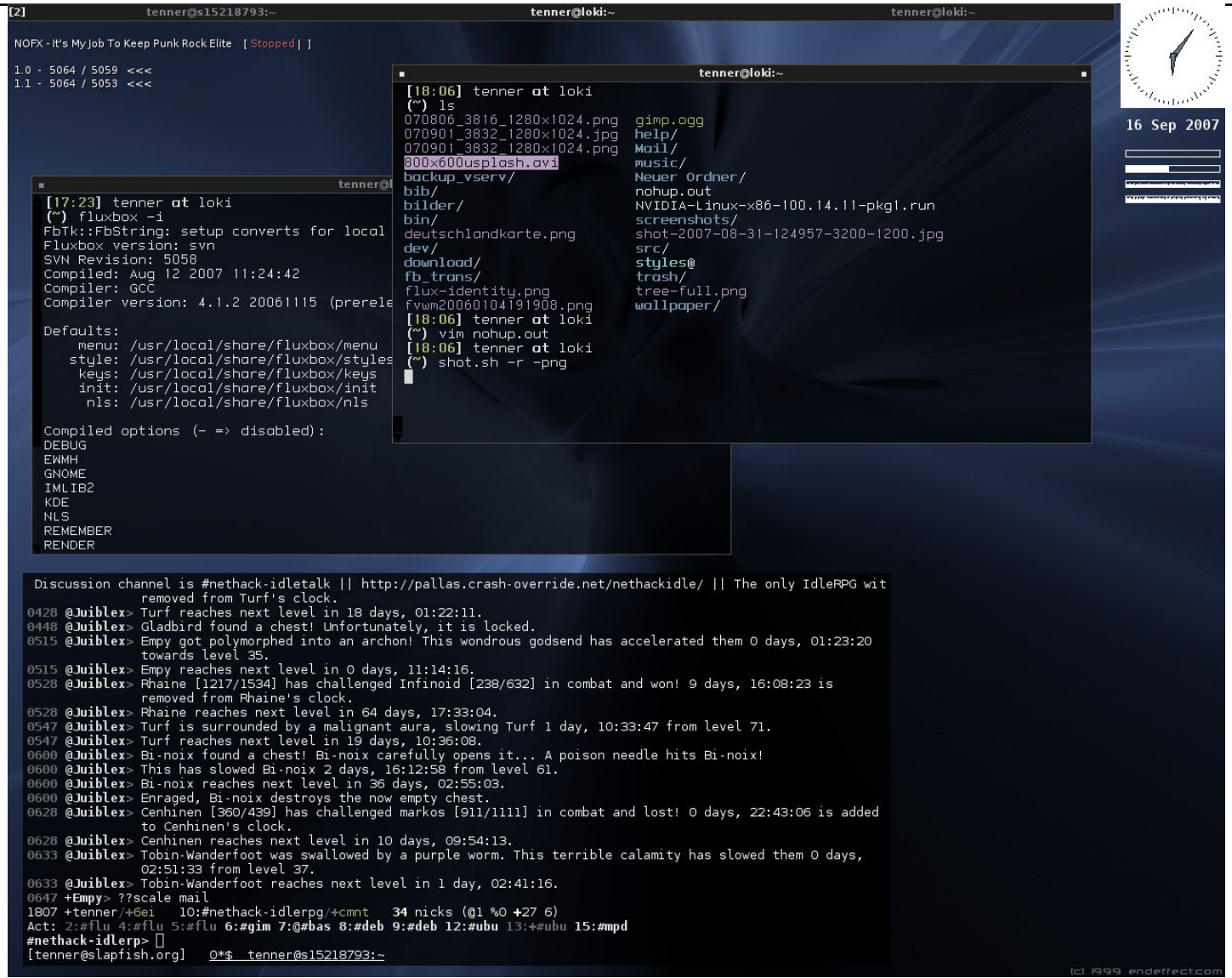
```
sudo yum groupinstall xfce -y
```

^---- if you did a minimal install of CentOS (no graphics!)

Make default on boot (if you want)

```
sudo systemctl set-default graphical.target
```

Fluxbox


[2] tenner@s15218793:~ tenner@loki:~ tenner@loki:~
NOFX - It's My Job To Keep Punk Rock Elite [Stopped |]
1.0 - 5064 / 5059 <<<
1.1 - 5064 / 5053 <<<
tenner@loki:~
[18:06] tenner at loki
(~) ls
070806_3816_1280x1024.png
070901_3832_1280x1024.jpg
070901_3832_1280x1024.png
800x600usplash.avi
backup_vserv/
bib/
bilder/
bin/
deutschlandkarte.png
dev/
download/
fb_trans/
flux-identity.png
fvwm20060104191908.png
gimp.ogg
help/
Mail/
music/
Neuer Ordner/
nohup.out
NVIDIA-Linux-x86-100.14.11-pkg1.run
screenshots/
shot-2007-08-31-124957-3200-1200.jpg
src/
styles@
trash/
tree-full.png
wallpaper/
[18:06] tenner at loki
(~) vim nohup.out
[18:06] tenner at loki
(~) shot.sh -r -png
[17:23] tenner at loki
(~) fluxbox -i
FbTk::FbString: setup converts for local
Fluxbox version: svn
SVN Revision: 5058
Compiled: Aug 12 2007 11:24:42
Compiler: GCC
Compiler version: 4.1.2 20061115 (prerele
Defaults:
menu: /usr/local/share/fluxbox/menu
style: /usr/local/share/fluxbox/styles
keys: /usr/local/share/fluxbox/keys
init: /usr/local/share/fluxbox/init
nls: /usr/local/share/fluxbox/nls
Compiled options (- => disabled):
DEBUG
EWMH
GNOME
IMLIB2
KDE
NLS
REMEMBER
RENDER
16 Sep 2007
Discussion channel is #nethack-idletalk || http://pallas.crash-override.net/nethackidle/ || The only IdleRPG wit
removed from Turf's clock.
0428 @Juiblex> Turf reaches next level in 18 days, 01:22:11.
0448 @Juiblex> Gladbird found a chest! Unfortunately, it is locked.
0515 @Juiblex> Empy got polymorphed into an archon! This wondrous godsend has accelerated them 0 days, 01:23:20
towards level 35.
0515 @Juiblex> Empy reaches next level in 0 days, 11:14:16.
0528 @Juiblex> Rhaine [1217/1534] has challenged Infinoid [238/632] in combat and won! 9 days, 16:08:23 is
removed from Rhaine's clock.
0528 @Juiblex> Rhaine reaches next level in 64 days, 17:33:04.
0547 @Juiblex> Turf is surrounded by a malignant aura, slowing Turf 1 day, 10:33:47 from level 71.
0547 @Juiblex> Turf reaches next level in 19 days, 10:36:08.
0600 @Juiblex> Bi-noix found a chest! Bi-noix carefully opens it... A poison needle hits Bi-noix!
0600 @Juiblex> This has slowed Bi-noix 2 days, 16:12:58 from level 61.
0600 @Juiblex> Bi-noix reaches next level in 36 days, 02:55:03.
0600 @Juiblex> Enraged, Bi-noix destroys the now empty chest.
0628 @Juiblex> Cenhinen [360/439] has challenged markos [911/1111] in combat and lost! 0 days, 22:43:06 is added
to Cenhinen's clock.
0628 @Juiblex> Cenhinen reaches next level in 10 days, 09:54:13.
0633 @Juiblex> Tobin-Wanderfoot was swallowed by a purple worm. This terrible calamity has slowed them 0 days,
02:51:33 from level 37.
0633 @Juiblex> Tobin-Wanderfoot reaches next level in 1 day, 02:41:16.
0647 +Empy> ??scale mail
1807 +tenner/+6ei 10:#nethack-idlerpg/+cmnt 34 nicks (@1 %0 +27 6)
Act: 2:#flu 4:#flu 5:#flu 6:#gim 7:@#bas 8:#deb 9:#deb 12:#ubu 13:+#ubu 15:#mpd
#nethack-idlerp>
[tenner@slapfish.org] 0*$ tenner@s15218793:~
(c) 1999 endeffect.com

Windows Managers and Desktop Environments

Window Manager	Description
Compiz Fusion	A highly configurable and expandable window manager based on the Compiz and Beryl window managers that utilizes 3D acceleration on modern video cards to produce 3D graphical effects, including 3D window behavior and desktop cube workspaces.
enlightenment	A highly configurable window manager that allows for multiple desktops with different settings. It is commonly used by the GNOME desktop.
fvwm	A window manager based on twm that uses less computer memory and gives the desktop a 3-D look; its full name is "Feeble Virtual Window Manager."
kwin	The window manager used for the KDE desktop.
lxde	A window manager specifically designed for use on underpowered systems such as netbooks, mobile devices, and legacy computers; its full name is "Lightweight X Desktop Environment."
metacity	The default window manager used by the GNOME version 1 and 2 desktop environments.
mutter	The default window manager used by the GNOME version 3 desktop environment.
mwm	A basic window manager that allows the user to configure settings using standard X utilities; its full name is "Motif Window Manager."
sawfish	A window manager commonly used for the GNOME desktop. It allows the user to configure most of its settings via tools or scripts.
twm	One of the oldest and most basic window managers; its full name is "Tab Window Manager."
wmaker	A window manager that provides drag-and-drop mouse movement and imitates the NeXTSTEP operating system interface made by Apple Computer, Inc.; its full name is "Window Maker."

Table 8-4: Common window managers

Windows Managers and Desktop Environments

- K Windows Manager (kwm)
  - Window manager that works under KDE
- Qt toolkit
  - Software toolkit included in KDE
- GNOME desktop environment:
  - Default desktop environment in most Linux distributions
  - Uses GTK+ toolkit
- Can configure KDE or GNOME to use different window manager
  - e.g., Compiz Fusion

Starting and Stopping X Windows

- When the init daemon boots to runlevel 5 or graphical.target, GNOME Display Manager (GDM) starts
  - Displays graphical login screen
  - Allows user to choose the desktop environment

- If you use runlevel1 (or rescue.target) or runlevel 2-4, the GDM is not started by default
  - Type **startx** at a character terminal to start X Windows and the default window manager

Configuring X Windows

- X Windows interfaces with video hardware
  - Requires information regarding keyboard, mouse, monitor, and video adapter card
- Attempts to automatically detect required information
  - If automatic detection fails, user needs to specify correct hardware information manually

Configuring X Windows

- User-configured settings are stored in files under the /etc/X11/xorg.conf.d directory
- Common settings such as the display resolution can be modified using the Displays utility within the GNOME desktop environment
  - Or by hand…

Configuring X Windows

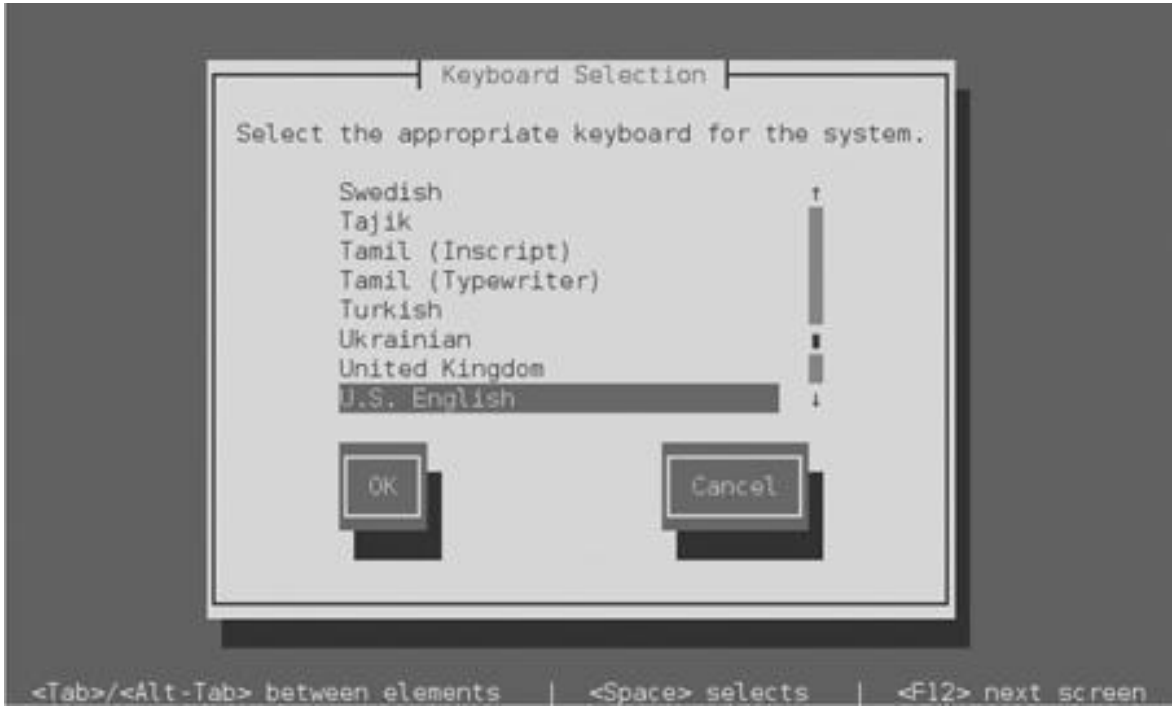


Figure 8-11: Selecting a keyboard layout

<Insert Linux Pun>