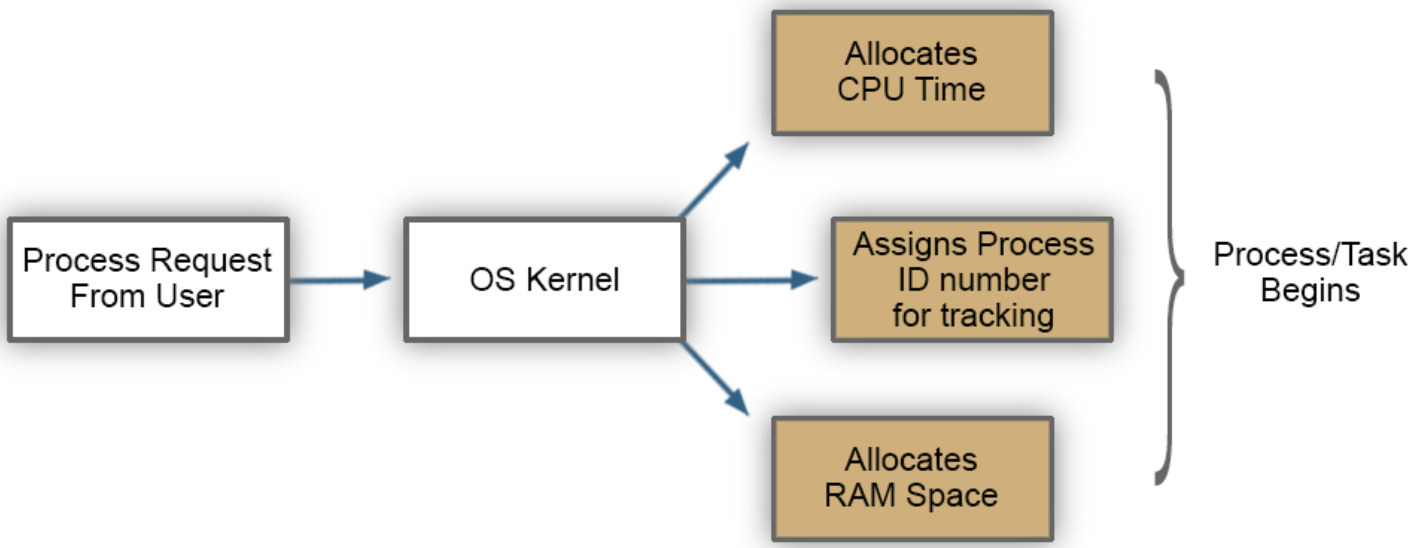
Chapter 11: Managing Processes

Process Initiation and Tracking



Process Control

- Linux manages tasks using processes
- Processes can be initiated by either the OS or by users
- A process can start a subprocess and form a parent/child relationship
- User's can only control their processes
- Root can control all system and user processes

The ps command

- The `ps` (process status) command lists running processes
- By itself, lists processes running in the current terminal:

```
$ps
PID TTY TIME CMD
80 ? 00:00:00 bash
94 ? 00:00:00 ps
```

PID	Process identifier unique to each process.
TTY	Name of the terminal or pseudo-terminal where the process is running.
TIME	Total processor time used by the process.
CMD	Command that started the process.

The ps command

- Use –`e` (every) and –`f` (full) to list all processes on the system:

$ps -ef

UID	PID	PPID	C	STIME	TTY	TIME	CMD
root	1	0	0	17:16	?	00:00:00	/sbin??? /init
syslog	33	1	0	17:16	?	00:00:00	/usr/sbin/rsyslogd
root	38	1	0	17:16	?	00:00:00	/usr/sbin/cron
root	40	1	0	17:16	?	00:00:00	/usr/sbin/sshd
bind	57	1	0	17:16	?	00:00:00	/usr/sbin/named -u bind
root	70	1	0	17:16	?	00:00:00	/bin/login -f
sysadmin	80	70	0	17:16	?	00:00:00	-bash
sysadmin	96	80	0	17:26	?	00:00:00	

Foreground Processes

- A foreground process is one that prevents the user from using the shell until the process is complete.
- When one process starts another, the first process referred to as the *parent process* and the new process is called a *child process*.

Executing Multiple Commands

- It can be useful to execute two or more commands in a single command line
- Syntax: cmd1 ; cmd2
- Example:

  $cd;clear
- Useful for creating an alias that runs multiple commands

Background Processes

- When executed in the background, a child process releases control back to the parent process
- To have a command execute as a background process, add the "&" after the command

```
$sort largefile &
$
```

Managing Jobs

- The `jobs` command displays background jobs:

```
$jobs
[1]-  Stopped          sleep 1000
[2]+  Stopped          sleep 2000
```

- The fg (foreground) and bg (background) commands provide the ability to multi-task:

```
$bg 1
[1]-  sleep 1000 &
$fg 2
sleep 2000
```

Signals

- A signal is a message that is sent to a process to tell it to take some sort of action, such as stop, restart, or pause
- Some signals can be sent to processes by simple keyboard combinations:

  $CTRL+z

  $CTRL+c

Signals

- The `bg` command sends a process a signal to execute in the background
- To list all signals, use the `kill -l` command
- To send a process a signal, use the kill command followed by the PID# or Job# (%x):

  $kill 2901

  $kill %1

  $ kill -p 2901

Force Kill

- If other signals have failed to end a process, use the SIGKILL signal to force the process to end:

  $kill -9 2901
  $kill -KILL %1
  $kill -SIGKILL -p 2901

Other Signal Commands

- There are other commands that send processes signals such, as the `killall` and `pkill` commands
- They are useful to stop many processes at once
- To stop all processes owned by a user:

```
$killall -u bob
```

The HUP Signal

- When a user logs off the system, all processes that are owned by that user are automatically sent the Hang-up signal (SIGHUP)
- Typically, this signal causes those processes to end
- To have a process ignore HUP signals:

```
$nohup myjob.sh &
```

Process Priority

- Not all processes have the same access to the CPU
- A user of can influence the priority that will be assigned to a process by setting a *niceness* value
- The higher you set the niceness value, the lower the priority that will be assigned to a process
- Highest: -20 Default: 0 Lowest: 19

Process Priority

- To set an initial niceness of a command, use the `nice` command:

```
$nice -n 19 cat /dev/zero > /dev/null
```

- To adjust the niceness of an existing process, use the renice command
- Only the root user can adjust a nice value to below 0 or lower than current value

The top Command

- Monitors processes in real time using the full terminal
  - Press “h” to see all available options
  - Press “r” for renice
  - Press “q” to quit

The uptime command

- The `uptime` command displays:
  - The current time
  - The amount of time the system has been running,
  - The number of users who are currently logged in
  - The *load averages* during the past one, five and fifteen minute.

The free command

- The `free` command displays:
  - Total amount of memory
  - Memory used
  - Free memory

Additional tools

- The `gnome-system-monitor` provides four tabs to view information about the System, Processes, Resources and File Systems