PRONET

VNU-ITP

UNDP

Processes Management

Objectives

PRONET

VNU-ITP

UNDP

- Define and describe various Linux *processes* and *daemons*
- Define and analyze process control
- Assign process *priority*
- Explain process *scheduling* and process *accounting*

Programs and Processes

- A *program* is an executable file, for example: /bin/bash
- A *process* is an *instance* of a program being executed. Process is also called *task*
- The kernel arranges for each process to have a share of the CPU and automatically switches from executing one process to executing another

Programs and Processes

- Every process has a priority associated with it
- Each time a context switch is made (i.e., a new process is to be run) the kernel searches a queue of these processes and selects the one with the *highest priority*
- A user or programmer can *reduce* a process's *priority* using the `nice` or `renice` command but *only the super user may increase the priority*

System Process Overview

- Five types of processes on Linux system:
  - Daemon
  - Parent
  - Child
  - Orphan
  - Zombie or defunct

Daemons and Zombies

- Daemon processes are Linux processes that support system functions.
- The key attribute of a daemon process is the fact that it is not attached to a terminal; this is shown on a `ps` listing as a question mark (?) in the terminal (TTY) column
- Daemons typically *wait for an event* such as a signal, a file being created, data input from a serial line or network, or a time out. *When the event occurs, the daemon wakes up, services the event, then goes to sleep again*

Daemons and Zombies

- Zombie processes are those processes that have finished but the parent process does not clean it up
- The only way to remove zombies in this situation is to terminate the parent process (see the kill command) and restart the application

Key Attributes of a Process

- Every process is identified by a unique number, process ID (PID)
- each process has an owner and a group: the effective UID and GID, that define the processes' access to the system

Key Attributes of a Process

- The kernel maintains a large amount of information about each process :
  - **Process ID (PID):** A positive integer value that uniquely identifies the process in the system ,PIDs are allocated in sequence automatically
  - **Text, data, and stack segments :** process's virtual memory
  - **User and Group IDs**

Checking on Processes

- Command Format :

```
ps [-options]
```

Options :

`-e`	Print information about *every* process on the system.
`-f`	Generate a *full listing*.
`-l`	Generate a *long listing*

PRONET

VNU-ITP

UNDP

```
$ ps -ef
```

UID	PID	PPID	C	STIME	TTY	TIME	CMD
root	1	0	80	16:46:44	?	0:40	/etc/init
root	2	0	27	16:46:44	?	0:00	pageout
aster	1292	1	80	06:48:51	console	0:01	-ksh
henry	231	1	80	06:48:51	pts/1	0:01	bash

Searching for a Specific Process

PRONET

VNU-ITP

UNDP

```
$ ps -e | grep lp
225     ?     0:01 lpNet
217     ?     0:0  lpsched
260     ?     0:01 lpNet
```

```
$ pgrep -l lp
225 lpNet
217 lpsched
260 lpNet
```

`top` Command

- The `top` provides an ongoing look at processor activity in real time
- It displays a listing of the most CPU-intensive *tasks* on the system, and can provide an interactive interface for manipulating processes: sort the tasks by CPU usage, memory usage, and runtime
- Both Gnome and KDE have GUI versions of top: `gtop` and `ktop` respectively. Users familiar with the Windows Task Manager will recognize `ktop` as a clone of this Windows feature

Process Priority and Nice Number

- When assigning *priority* to a process, the kernel supplies something called a *nice number*. Nice numbers range from -20 (highest) to +19 (lowest). Processes started by a user have a nice value of 0

`nice` Command

- The `nice` command to modify the nice numbers on processes *at start time*
- Regular users can only adjust nice values from 0 to 19, while super users can adjust nice values from -20 to 19. If no adjustment value is given, then the process is assigned a value of 10
- If no commands are specified, then `nice` displays the current scheduling priorities

`nice` Command

PRONET VNU-ITP UNDP

- Command format :

```
nice [options] [command [arg]…]
```

- Examples:

`# nice -2 netscape`	Set nice number is 2
`# nice --2 netscape`	Set nice number is -2
`# nice -n -2 netscape`	Set nice number is -2

`renice` Command

- The `renice` command to modify the nice numbers on processes that *are already running*
- There is some syntactical difference between nice and renice, mainly that renice does not require the use of two dashes (i.e. `--2`) to signify a negative nice number

`renice` Command

- Command format :

```
renice priority PID [[-g] group] [[-u] user]
```

- Examples:

```
# renice -2 203
```

Set nice number is -2 to PID=203

```
# renice 5 -u henry
```

Set nice number is 5 to all processes started by henry

Signals

- Hardware-related program faults (Divide by zero), press Ctrl+C, … can cause signals
- System administrators usually *use signals to manipulate processes*.
- There are a number of standard signals. You can use `kill -l` or `man 7 signal` to list all signals supported by the system

Signals

PRONET VNU-ITP UNDP

- (refer to `$ man 7 signal`)

Name	Value	Comment
SIGHUP	1	Hangup detected on controlling terminal or death of controlling process
SIGINT	2	Interrupt from keyboard (Ctrl+C pressed)
SIGKILL	9	Kill signal
SIGTERM	15	Terminate program

...

Sending Signals to Processes

- The `kill` and `pkill` commands can help you to send signals to processes

`kill` Command

PRONET VNU-ITP UNDP

- Provides a direct way to terminate unwanted command processes
- Command Format

  `kill` `[-signal] <PID(s)>`

- Terminating a Process :

  1. Type **`ps`** to find out the PIDs for the processes.
  2. Type **`kill`** followed by the PIDs.

Finding and Terminating a Process

PRONET

VNU-ITP

UNDP

```
$ ps -u henry
PID      TTY      TIME     COMD
12892    console  0:01     ksh
12932    pts/0    0:01     find
12935    pts/1    0:00     ps

$ kill 12932
or
$ kill -9 <PID>
```

`pkill` Command

PRONET

VNU-ITP

UNDP

- Command Format

  pkill [*-signal*] [*opt*] <*pattern*>

Option :

```
  -u      : processes matched with UID
  -x      : match exactly
```

Intercepting Signals

- The shell can register signal handlers using the `trap` command. Traps are used mostly within programs and scripts:

```
trap ["[action]"] signals
```

`action`	List of *command(s*), separated by ';' and *must be placed in double quotes*
`signals`	List of signal name(s) or signal number(s)

Intercepting Signals Examples

PRONET

VNU-ITP

UNDP

- Set traps :

```
# trap "echo interrupt ignored" INT QUIT

# trap "echo interrupt ignored" 2 3

# trap "" 2 QUIT
```

- Reset traps :

```
# trap 2 3
```

Scheduling Processes

PRONET

VNU-ITP

UNDP

- The `at` command
- The `crontab` command

`at` Command

- Execute commands *at a specified time* or run the commands *on a batch queue*.

```
at [options] <time> <date>
```

- `at` reads commands from `stdin` or file (with `-f` option) and executes them using user'shell
- `stdout` and `stderr` of this command *are mailed to the user* who run the command by default
- (Examples)

`at` Examples

- Run *myprogram* once at 6:15 p.m tomorrow:

```
$ at 6:15pm tomorrow
at> myprogram
at> ^D
```

- The ^D indicates that the user typed Ctrl-D on the keyboard, sending the end-of-file character to terminate the at command

`at` Examples

PRONET VNU-ITP UNDP

- Run commands that are listed in the file *command_list* at 9 p.m. two days from now:

```
$ at -f command_list 9pm + 2 days
```

Listing and Deleting `at` Jobs

PRONET VNU-ITP UNDP

`at -l`	List jobs, it's an alias for `atq`
`at -d` *`job_id`*	Removes jobs, it's an alias for `atrm`
`atq`	Like `at -l`
`atrm` *`job_id`*	Like `at -d`

```
# atq
14      2003-10-31 12:00 a root
# atrm 14
# atq
```

The `crontab` Command

PRONET VNU-ITP UNDP

- Use `crontab` to run jobs at *periodic intervals*
- List cron table with:

```
crontab [-u user] -l
```

- Edit the current cron table with:

```
crontab [-u user] -e
```

- Remove the cron table with:

```
crontab [-u user] -r
```

crontab File Structure

PRONET VNU-ITP UNDP

- Each line in in a crontable includes commands and information to determine when to run the command :

```
<minute> <hour> <day> <month>  <weekday> <command>
```

  - Space/tab separated fields (columns)
  - Use comma-separated list or range of values in each field

`minute`	Minute (0 through 59)
`hour`	Hour (0 through 23)
`day`	Day of the month (1 through 31)
`month`	Month (1 through 12 or Jan through Dec)
`weekday`	Day of the week (0 through 6 [where 0 is Sunday] or sun through sat)
`command`	Command (any valid command, including spaces and standard Bourne shell syntax)

crontab File Structure

PRONET

VNU-ITP

UNDP

- Fields with '*' will match any values
- Lines that begin with the pound sign (#) are *comment lines* and are ignored by **crond**
- For more details about file structure:

```
#man 5 crontab
```

crontab File

- Typically, system administrators use crontab to *tidy up* the system (truncate log files, remove out-of-date or unused files, etc.) and *perform regular operations* such as file system backups or accounting programs
- Each user has a personal list of commands kept in the crontab file, stored in `/var/spool/cron/` directory

Examples

PRONET VNU-ITP UNDP

- To execute *myprogram* once per day at 6:15 a.m , use this **crontab** entry:

```
15 6 * * * myprogram
```

- To execute at 6:15 and 18:15 on the 1st and 15th of the month, use:

```
15 6,18 1,15 * * myprogram
```

Examples

PRONET

VNU-ITP

UNDP

- **User may use crontab to set regular reminders:**

```
0 8 * * 5 echo "Fill in timesheet
                                today"

0 8 29 2 * echo "Happy Birthday"

11 1 * * 0 /home/bin/full.backup

11 1,13 * * 1-5 /home/bin/inc.backup
```

Administering `at` and `crontab`

PRONET VNU-ITP UNDP

- Control and configuration files are kept in /etc

`at.allow`	Users allowed to use `at`
`at.deny`	Users denied use of `at` (only used if no `at.allow`)
`cron.allow`	Users allowed to use `cron`
`cron.deny`	Users denied use of `cron`
`crontab`	System `cron` table

Administering `at` and `crontab`

PRONET VNU-ITP UNDP

- These files are simply lists of account names.
- If the *allow* file exists, only those users listed in the *allow* file may use the service. If the *allow* file does not exist but the *deny* file does, only those users not listed in the *deny* file may use the service.
- For **cron**, if neither file exists, all users have access to **cron**.
- For **at**, if neither file exists, only root has access to **at**. An empty *at.deny* file allows access to all users and is the default.

Administering `at` and `crontab`

PRONET
VNU-ITP
UNDP

- All crontabs are stored in the cron spool directory `/var/spool/cron`
- All at jobs are stored in the cron spool directory `/var/spool/at`
- Do not edit the files in the spool directories. Use the `at` and `crontab` utilities to make changes

Summary

PRONET

VNU-ITP

UNDP

- Define and describe various Linux *processes* and *daemons*
- Define and analyze process control
- Assign process *priority*
- Explain process *scheduling* and process *accounting*