Process Control

- processes and executables
- job control
- ps and kill
- top
- at
- cron, crontab, and calendar

Processes vs. Executables

- A process is a program that is executing in memory.
- An executable is a program that resides in the filesystem.

Foreground and Background Processes

- A process is in the foreground when the shell waits for the process to complete before executing another command typed by the user. You can interact with a foreground process in a shell (e.g. receiving input from the keyboard and/or writing to the screen).
- A process is in the background when the shell does not wait for the process to complete.

Stopping a Foreground Process

- A user can kill a foreground process by typing CTRL-c in the shell window running that process.
- A user can stop a foreground process by typing CTRL-z in the shell window running that process. The process remains in a suspended state until it is killed, put in the background, or put back in the foreground.

Job Control (23)

- The management and manipulation of foreground and background processes is called job control. A job is another name for a background or suspended process and is identified by a job number.
- You can list the background and stopped jobs associated with the current terminal identifier with their job numbers using the *jobs* command.

Putting a Process in the Background

- A job can be put in the background by initiating the command with a & following it.
- You can also put a stopped process in the background by using the *bg* command. If the job number is not specified, then the most recently stopped job is put in the background.

  bg [%*num*]

Putting a Process in the Foreground

- One can use the *fg* command to put a suspended or background job in the foreground. If the job number is not specified, then the most recently stopped job is put in the foreground.

  fg [%*num*]

Ps: Reporting Process Status

- The *ps* command prints information about current processes on a machine. The process ID (PID), terminal identifier, cumulative execution time, and command name are given. By default, only the processes associated with a terminal are shown. The PID is guaranteed to be unique among current processes for that processor.
- General form.

  ps [*options*]

Commonly Used Ps Options

-a # lists information about all processes associated
with any terminal
-l # lists processes in a long listing (more information)
-u *uidlist* # lists processes associated with the list of user ID
numbers or logins

Suspending a Job

- A user can suspend a background job or a process by using the *stop* command. Either the PID or the job number can be specified as an argument.

  stop [%*num* | *pid*]

- A process can suspend itself for a specified amount of time by issuing the *sleep* command.

  sleep *seconds*

Terminating a Process

- A process can be terminated by using the *kill* command. Either the job number or PID can be specified.
- General form. One common option is -9 which represents a signal number representing SIGKILL to unconditionally terminate a process.

  kill [options] %*num* # kill a process by job number

  kill [options] *pid* # kill a process by PID

Top: Display Information about the Top CPU Processes

- The *top* unix utility displays information about the most active processes on the processor and periodically updates this information. The percentage of CPU use is used to rank the processes. *Top* is useful to determine which processes appear to be slowing down a processor.
- General form:

  top [*options*]

Some Commonly Used Top Options

-d*count* # only show *count* updates and then exit
-I # do not display idle processes
-s*time* # set time between updates (default is 5 seconds)
-U*username* # show only the processes owned by *username*

Some Top Commands

h # display a summary of the commands
q # quit top
d # will prompt for count and display that count of
  # updates before termination
n # will prompt for the number of processes and display
  # only that number of processes
s # will prompt for the number of seconds between
  # displays
u # will prompt for a username and display only those
  # processes owned by that user

At: Execute Commands at a Later Time

- The *at* unix utility reads commands from standard input and groups them together to be executed at a specified time.
- General form. The *timespec* includes a time and an optional date.

% at [*options*] *timespec* # specify time the cmds are performed
% at> *command1* # first command
% at> ... # ...
% at> *commandn* # last command
% at> CTRL-d # end of input
%

Example Usage of the At Command

- The -m option indicates to send the standard output as an e-mail message to the user when the commands are performed.

% at -m 15:45
% at> echo “Go to department faculty meeting in”
% at> echo “room 151 Love at 4pm.”
% at> CTRL-d
%

Atq and Atrm Commands

- One can use the *atq* command to get a list of the pending *at* jobs and when they will be initiated.

  atq

- One can use the *atrm* command to remove a particular *at* job.

  atrm *atjobnum*

Cron and Crontab

- *Cron* starts a process that executes commands at specified dates and times.
- One easy interface is to perform a “crontab -e” to edit the *crontab* file. This file consists of lines consisting of five fields indicating when a command is to be performed and a sixth field containing the command.
- The five integer fields represent the time in the crontab file:

  minute (0-59), hour (0-23), day of month (1-31), month of year (1-12), day of week (0-6 with 0=Sunday)

Using Cron with the Calendar Program

- The *calendar* utility consults the file “calendar” in the current directory and writes lines that contains today's or tomorrow's (or next weekday's) date anywhere in the line to standard output.
- Below is a line you can put in the *crontab* file using the “crontab -e” command to send e-mail containing the output of the *calendar* program at 3am. The *'s indicate that this command should occur every day.

  0 3 * * * calendar | mailx whalley@cs.fsu.edu

Using Cron with the Calendar Program

- The *calendar* utility consults the file “calendar” in the current directory and writes lines that contains today's or tomorrow's (or next weekday's) date anywhere in the line to standard output.
- Below is a line you can put in the *crontab* file using the “crontab -e” command to send e-mail containing the output of the *calendar* program at 3am. The *'s indicate that this command should occur every day.

  0 3 * * * calendar | mailx whalley@cs.fsu.edu