Job Control

CIS 2235 Linux System Administration

Review - What is a Process?

The kernel considers each program running on your system to be a process.

A process 'lives' as it executes, with a lifetime that may be short or long. Processes consume resources.

A process is said to 'die' when it terminates.

The kernel identifies each process by a number known as a process ID, or **pid**.

The kernel keeps track of various properties of each process.

Process properties

A process has a user id (**uid**) and a group id (**gid**), which specify its permissions.

A process has a parent process id (**ppid**) — the pid of the process which created it.

The kernel starts an initial process with pid 1 at boot-up.

Most other processes are a descendant of pid 1.

Each process has its working directory, initially inherited from its parent process..

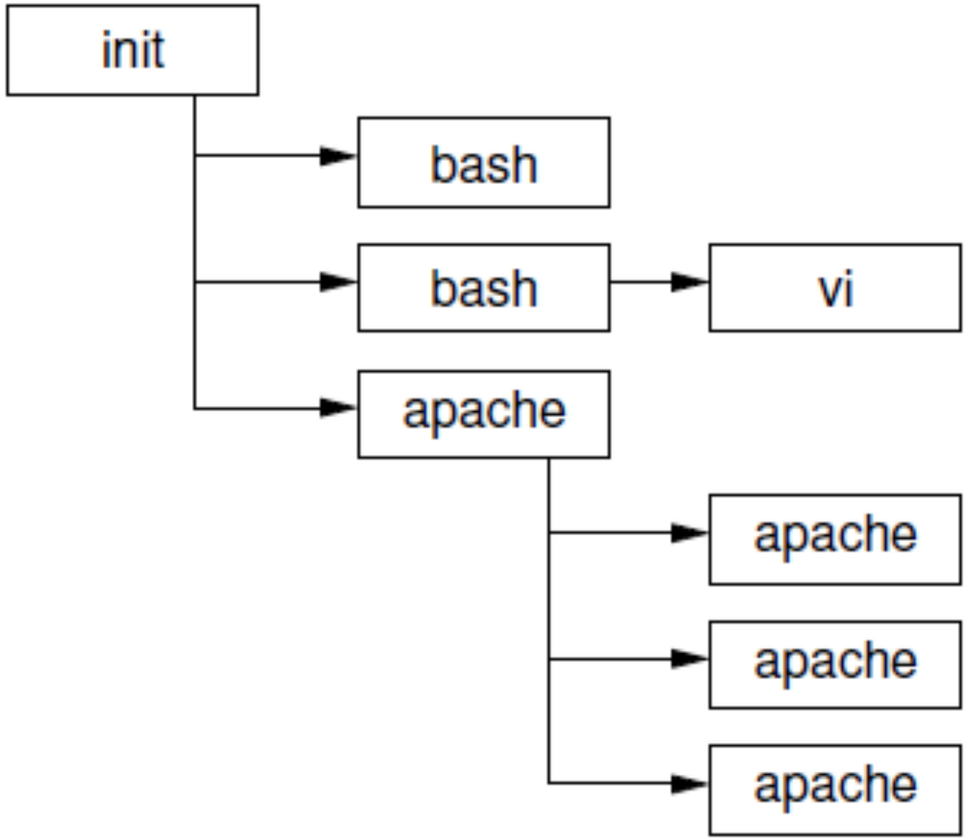
Each process has an environment — a collection of named environment variables and their associated values.

A process's environment is usually inherited from its parent process.

The kernel also tracks the memory a process uses, what files and network connections it has open, and information about how much execution time the process has consumed.

init

The `init` process is the ancestor of most other processes



PID/PPID Example

```
ldamon@ubuntuLTS:~$ ps -elf
F S UID          PID  PPID  C PRI  NI ADDR SZ WCHAN  STIME TTY          TIME CMD
4 S root           1     0  0  80   0 -  9489 -      Feb20 ?        00:00:02 /sbin/init
1 S root           2     0  0  80   0 -     0 -      Feb20 ?        00:00:00 [kthreadd]
4 S root        1055     1  0  80   0 - 16381 -      Feb20 ?        00:00:00 /usr/sbin/sshd -D
4 S root        1192  1055  0  80   0 - 23851 -      Feb20 ?        00:00:00 sshd: ldamon [priv]
4 S root        7603  1055  0  80   0 - 23851 -      15:33 ?        00:00:00 sshd: ldamon [priv]
5 R ldamon      7675  7603  0  80   0 - 23851 -      15:33 ?        00:00:00 sshd: ldamon@pts/1
0 S ldamon      7676  7675  0  80   0 -  5647 wait   15:33 pts/1    00:00:00 -bash
0 R ldamon      7698  7676  0  80   0 -  9342 -      15:33 pts/1    00:00:00 ps -elf
```

Getting the environment for a process

```
ldamon@ubuntuLTS:~$ ps ewww
  PID TTY      STAT   TIME COMMAND
 7736 pts/1    R+     0:00 ps ewww XDG_SESSION_ID=1797 TERM=xterm-256color SHELL=/bin/bash
SSH_CLIENT=10.0.2.2 55525 22 SSH_TTY=/dev/pts/1 USER=ldamon
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd=40;33;01:or=40;31
;01:mi=00:su=37;41:sg=30;43:ca=30;41:tw=30;42:ow=34;42:st=37;44:ex=01;32:*.tar=01;31:*.tgz=01;31:*
.arc=01;31:*.arj=01;31:*.taz=01;31:*.lha=01;31:*.lz4=01;31:*.lzh=01;31:*.lzma=01;31:*.tlz=01;31:*.
txz=01;31:*.tzo=01;31:*.t7z=01;31:*.zip=01;31:*.z=01;31:*.Z=01;31:*.dz=01;31:*.gz=01;31:*.lrz=01;3
1:*.lz=01;31:*.lzo=01;31:*.xz=01;31:*.bz2=01;31:*.bz=01;31:*.tbz=01;31:*.tbz2=01;31:*.tz=01;31:*.d
eb=01;31:*.rpm=01;31:*.jar=01;31:*.war=01;31:*.ear=01;31:*.sar=01;31:*.rar=01;31:*.alz=01;31:*.ace
=01;31:*.zoo=01;31:*.cpio=01;31:*.7z=01;31:*.rz=01;31:*.cab=01;31:*.jpg=01;35:*.jpeg=01;35:*.gif=0
1;35:*.bmp=01;35:*.pbm=01;35:*.pgm=01;35:*.ppm=01;35:*.tga=01;35:*.xbm=01;35:*.xpm=01;35:*.tif=01;
35:*.tiff=01;35:*.png=01;35:*.svg=01;35:*.svgz=01;35:*.mng=01;35:*.pcx=01;35:*.mov=01;35:*.mpg=01;
35:*.mpeg=01;35:*.m2v=01;35:*.mkv=01;35:*.webm=01;35:*.ogm=01;35:*.mp4=01;35:*.m4v=01;35:*.mp4v=01
;35:*.vob=01;35:*.qt=01;35:*.nuv=01;35:*.wmv=01;35:*.asf=01;35:*.rm=01;35:*.rmvb=01;35:*.flc=01;35
:*.avi=01;35:*.fli=01;35:*.flv=01;35:*.gl=01;35:*.dl=01;35:*.xcf=01;35:*.xwd=01;35:*.yuv=01;35:*.c
gm=01;35:*.emf=01;35:*.ogv=01;35:*.ogx=01;35:*.aac=00;36:*.au=00;36:*.flac=00;36:*.m4a=00;36:*.mid
=00;36:*.midi=00;36:*.mka=00;36:*.mp3=00;36:*.mpc=00;36:*.ogg=00;36:*.ra=00;36:*.wav=00;36:*.oga=0
0;36:*.opus=00;36:*.spx=00;36:*.xspf=00;36: LIBVIRT_DEFAULT_URI=qemu:///system MAIL=/var/mail/
ldamon PATH=/home/ldamon/bin:/home/ldamon/.local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/
usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin PWD=/home/ldamon LANG=en_US.UTF-8 SHLVL=1
HOME=/home/ldamon LOGNAME=ldamon XDG_DATA_DIRS=/usr/local/share:/usr/share:/var/lib/snapd/desktop
SSH_CONNECTION=10.0.2.2 55525 10.0.2.15 22 LESSOPEN=| /usr/bin/lesspipe %s XDG_RUNTIME_DIR=/run/
user/1000 LESSCLOSE=/usr/bin/lesspipe %s %s _=/bin/ps
```

Some common ps columns

Field	Contents
USER	Username of the process's owner
PID	Process ID
%CPU	Percentage of the CPU this process is using
%MEM	Percentage of real memory this process is using
VSZ	Virtual size of the process
RSS	Resident set size (number of pages in memory)
TTY	Control terminal ID
STAT	Current process status: R = Runnable D = In uninterruptible sleep S = Sleeping (< 20 sec) T = Traced or stopped Z = Zombie Additional flags: W = Process is swapped out < = Process has higher than normal priority N = Process has lower than normal priority L = Some pages are locked in core s = Process is a session leader
TIME	CPU time the process has consumed
COMMAND	Command name and arguments[a]

a. Programs can modify this information, so it's not necessarily an accurate representation of the actual command line.

Job control

Most shells offer job control.

The ability to stop, restart, and background a running process.

The shell lets you put `&` on the end of a command line to start it in the background.

Or you can hit Ctrl+Z to suspend a running foreground job.

```
$ vncviewer localhost:2
Password:
^Z
$ bg
```

Suspended and backgrounded jobs are given “job numbers” by the shell.

Job control part 2

These “job numbers” can be given to shell job-control built-in commands.

Job-control commands include `jobs`, `fg`, and `bg`.

`jobs` command displays jobs in the current shell.

jobs command

```
ldamon@ubuntuLTS:~$ jobs
[1]+  Stopped                 vi xx.c
[2]-  Stopped                 vi yy.java
[3]   Running                 tail -f /var/log/kern.log &
ldamon@ubuntuLTS:~$
```

foreground

```
$ fg
```

Brings a backgrounded job into the foreground

Re-starts a suspended job, running it in the foreground

`$ fg %1` will foreground job number 1

fg with no arguments will operate on the current job

background

The bg command:

Re-starts a suspended job, running it in the background.

`bg %1` will background job number 1.

bg with no arguments will operate on the current job.

For example, after running gedit and suspending it with Ctrl+Z, use bg to start the job running again in the background.

process signaling

The kernel or another process can send a process a signal.

Two ways to specify a signal:

- as a small whole number.
- with a mnemonic name.

Signal names are all-capitals, like INT.

They are often written with SIG as part of the name: SIGINT.

Some signals are treated specially by the kernel; others have a conventional meaning.

There are about 30 signals available, not all of which are very useful.

Most common signals

# [b]	Name	Description	Default	Can catch?	Can block?	Dump core?
1	HUP	Hangup	Terminate	Yes	Yes	No
2	INT	Interrupt	Terminate	Yes	Yes	No
3	QUIT	Quit	Terminate	Yes	Yes	Yes
9	KILL	Kill	Terminate	No	No	No
10	BUS	Bus error	Terminate	Yes	Yes	Yes
11	SEGV	Segmentation fault	Terminate	Yes	Yes	Yes
15	TERM	Software termination	Terminate	Yes	Yes	No
17	STOP	Stop	Stop	No	No	No
18	TSTP	Keyboard stop	Stop	Yes	Yes	No
19	CONT	Continue after stop	Ignore	Yes	No	No
28	WINCH	Window changed	Ignore	Yes	Yes	No
30	USR1	User-defined #1	Terminate	Yes	Yes	No
31	USR2	User-defined #2	Terminate	Yes	Yes	No

a. A list of signal names and numbers is also available from the **bash** built-in command **kill -l**.
b. May vary on some systems. See **/usr/include/signal.h** or **man signal** for more information.

Using kill to signal

The `kill` command is used to send a signal to a process.

To stop a process:

```
$ kill -SIGTERM pid
```

or, to insist:

```
$ kill -9 pid
```

However, `kill` is not just to terminate a running process!

Using kill to signal

Use `kill -HUP pid` or `kill -s HUP pid` to send a SIGHUP to the process with that pid.

If you forget the signal name, kill will send a SIGTERM, a politer way of saying kill.

You can specify more than one pid in a single kill command.

The signal will be sent to all the processes.

You cannot kill processes that don't belong to you (unless you are root; the superuser can kill any process that is killable).

signaling daemons

On Unix systems, long-lived processes that provide some service are often referred to as dæmons.

Dæmons typically have a configuration file (usually under /etc) which affects their behavior.

Many dæmons read their configuration file only at startup.

If the configuration *changes*, you have to explicitly tell the dæmon by sending it a SIGHUP signal.

Typical use:

```
$ sudo kill -HUP <pid of crond>
```

This tells the crond to re-read its cron file