Hands On UNIX

AfNOGChix 2011
Blantyre, Malawi

AfNOG

Processes

- A running instance of a program is called a "process"
- Identified by a numeric **process id (pid)**

  unique while process is running; will be re-used some time after it terminates
- Has its **own private memory space**

  not accessible by other processes; not even other instances of the same program

AfNOG

What does UNIX give a process?

- A **table of environment variables**
  - just a bunch of **name=value settings**
  - kept in memory (process gets own private copy)
- A table of open files
  - 0: standard input
  - 1: standard output
  - 2: standard error
- A set of argument strings
  - e.g. what you put after the command name
- THAT'S ALL!!

AfNOG

The shell: a simple interface

- The shell **lets you start processes**

  and waits for them to finish, unless you run them in the "background"

- The shell **lets you set environment variables**
- The shell **lets you set up file descriptors**

  Normally stdin is connected to your keyboard and stdout/stderr to your screen, but you can override

- The shell **lets you pass arguments**

AfNOG

Shell expansion

- The **shell performs processing on your command line before starting the program**
- Splits line into words (**cmd, arg1, arg2**,…)
- **Searches for cmd in PATH** if required
- Performs various types of argument expansion

  See exercise

AfNOG

The shell itself runs as a process

- A shell **can start another shell**
- A shell **has its own environment**

  e.g. it uses the PATH setting to locate programs

  it copies the environment to its children
- A shell has stdin/stdout/stderr

  You can run a **non-interactive shell, i.e. a script**

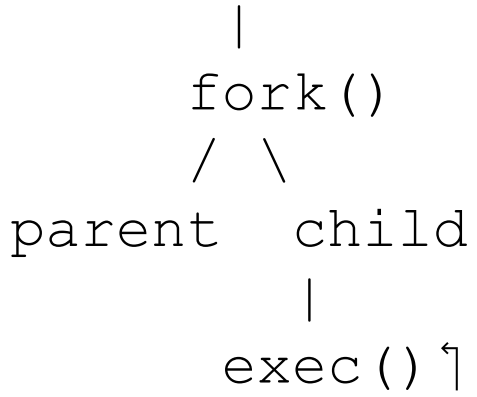
  Examples include periodic system tidying
  - log rotation
  - rebuilding of the locate database
  - rebuilding of the man page index

AfNOG

How are new processes started ?

- The **current processes "clones" itself via the fork() call**
- The **fork'ed copy is called the child**

  it shares all the characteristics of the parent, including memory, open files, etc...
- The **child replaces itself by calling the new program to run via exec()**



AfNOG

Once a process has started...

- It **can make "system calls" to the Kernel as needed**, e.g. to
  - read and write data
  - open and close files
  - start new child processes (known as "fork") ...etc
- **Using its pid, you can send it a "signal"**, e.g.
  - Request to **terminate**
  - Request to **suspend** (stop temporarily) or **restart**
  - Certain system events also send signals
- When it ends, **returns 'exit code' (0-127)**
  - to parent (the process which started it)

AfNOG

Process control from the shell

- For a **"foreground" process**

  Ctrl-C = terminate

  Ctrl-Z = suspend **

- Show all processes

  ps auxw

- Send a signal to any process

  kill [-sig] pid

- More advanced **job control**

  jobs = list all jobs (children) started by this shell

  fg %n = resume in foreground **

  bg %n = resume in background

AfNOG

Summary

- Processes **identified by pid**
- Each process at start gets 3 things:

  Environment variables, e.g. HOME="/home/you"

  Open files

  Arguments
- You can **send signals to a running process**
- At end it returns a numeric exit code
- Shell gives you control of these things

AfNOG

Processes and security

- Each process **runs with set privileges**

  effective uid

  effective gid

  supplementary groups

- Some operations are **only available to root**

  e.g. **bind socket** to port below 1024

  e.g. **shut down** system

- A process running as root **(euid=0**) can change to any other uid - but not back again

- **Other processes cannot change uid at all!**

AfNOG

How do users change passwords?

- Note that **/etc/master.passwd is only readable and writable by root**
- The '**passwd**' program has special privileges, it is marked "setuid root"
- Whenever a user starts the **'passwd' program, kernel gives it euid=root**

  It can then change the user's password
- setuid programs must be written very carefully to avoid security holes
- **Don't fiddle with setuid bits**

AfNOG

Aside...

- It's really useful to **think of commands in pairs**

  The command which **shows a setting** and the command which **changes that setting**

- Example:

  pwd shows the current working directory

  cd changes the current working directory

- Follow the 3-step system for changes

  Check things are how you think they are

  Make the change

  Check things have changed as you expected

AfNOG

The Virtual Filesystem (VFS)

- All filesystems appear in a single tree
- Must have a root device - /
- Can attach other devices at other points
- At **bootup, everything in /etc/fstab is mounted**

  except lines marked 'noauto'

AfNOG

Key VFS commands

- Show status

  mount

  df

- Attach device

  mount -t cd9660 /dev/acd0 /cdrom

  - **/cdrom** is called the "**mount point**"
  - it's just an empty subdirectory
  - after mounting, the filesystem contents appear here

- Detach device

  umount /cdrom

AfNOG

Other devices

- Formatting a floppy disk

  fdformat /dev/fd0

  newfs_msdos -L myfloppy /dev/fd0

- Mounting a floppy disk

  mount -t msdos /dev/fd0 /mnt

- USB pen

  mount -t msdos /dev/da0s1 /mnt

  - typical example
  - look in **/var/log/messages** to check device
  - use 'fdisk /dev/da0' to look at slices

AfNOG

Filesystem safety

- **DON'T remove any media until it has been unmounted**

  Otherwise, filesystem can be corrupted

- **Kernel won't let you unmount a filesystem if it is in use**

  Use 'fstat' to find processes using it

- A**LWAYS shut down properly**
- Filesystem repair tool is called "**fsck**"

AfNOG