PROCESSES

At least they're not ISO-9001 processes

STRUCTURE

- In Linux, a Process wraps up everything that is needed to know about a running piece of software
- The meta information not only includes the machine code for the software, but also things like what user/group pair is running the process, when it was started, what the command line was, etc.
- In fact, here's a short list of the pertinent parts of a process:

STRUCTURE

- PID
- PPID
- UID/GID
- Command
- Start Time
- CPU Time
- CWD
- State
- TTY
- Environment
- Priority
- Nice Level

PID

- Process ID
- Linux uses this number to uniquely identify every process on the computer
- Number from 1-32768 (default - can change the maximum)
- Assigns new PIDs incrementally by 1, 2 or 4
- Loops back to 1 after hitting the maximum

PPID

- Parent Process ID
- PID of the process that started this one
- What? Side track: The Fork & Exec model!

THE FORK AND EXEC MODEL

More whiteboard goodness!

Fork & Exec Model:

UID/GID

- The User and Group running the process
- Very important! Defines access and permissions to file system and operating system.
- Inherited from Parent process unless:
  - SetUID/SetGID bits on executable
- Completes the Circle of Security

COMMAND

- The command (and arguments) for the process
- Identifies the executable running, as well as the arguments passed at invocation

START & CPU TIME

- Start Time tracks when the process was started
- CPU Time tracks time the process actually spends **running on** the CPU

CWD

- Current Working Directory
- ‘nuf said
- Inherited from parent process

STATE

- State of the process:
  - Runnable
  - Stopped
  - Blocked - Interruptible
  - Blocked - Non-interruptible
  - Zombie

TTY

- Connected terminal
- Mostly informational
- Inherited from parent process

ENVIRONMENT

- Every process has it’s own Environment
- Inherited from parent process

PRIORITY

- The priority is a read-only value showing the current priority assigned by the scheduler
- Ranges from 0-99, with higher values representing higher priorities.
- The scheduler constantly adjusts priorities to balance efficiency, performance and responsiveness

NICE LEVEL

- The nice level represents one influence on the calculations the kernel uses when assigning priorities.
- Originally designed and named to allow users to be “nice” to other users of the system by assigning a higher nice value to an intensive process, which in turn lowers it’s priority.
- Ranges from -20 to 19. Default nice level is 0.
- Only root can assign negative nice values.
- See `nice` and `renice` commands

LISTING PROCESSES

- `ps`: List of current processes
- `pstree`: Generate hierarchical view of processes
- Examples:
  - `ps` *View all processes started by logged in user*
  - `ps aux` *View details of all processes on system*
  - `pstree` *View tree of all processes on system*

PROCESS STATES

- There are 5 basic process states:
  - Runnable
  - Stopped
  - Blocked/Sleeping - interrutible
  - Blocked/Sleeping - non-interrutible
  - Zombie/Defunct

RUNNABLE

- This means the process is running, or is set to run
- Remember: Linux is a multi-tasking operating system, so it's hard to see exactly when processes are running (switched so quickly), so the state is **runnable**, indicating that the scheduler will provide CPU time when it's available

STOPPED

- Opposite of Runnable - the process will not get CPU time
- Nothing happens to the process - it's still in memory, poised, ready to go. But when it's put in the stopped state, the scheduler will not put it on the CPU
- Files/network connections remain open, but network connections may drop after a time (timeout)

INTERRUPTIBLE SLEEP

- The process is waiting for some event - perhaps an alarm from a sleep system call, perhaps a signal or other external event

- Interruptible means that other processes/events can break the sleep

NON-INTERRUPTIBLE SLEEP

- This sleep state is generally caused by IO operations - accessing a drive, communicating with the network, etc.
- Non-interruptible means that other processes/events can not break this sleep.
- This process is unable to respond to signals.

ZOMBIE/DEFUNCT

- Braaaaaaiiiiiiinnnnnnssss.. Wait, no, not that kind of zombie.
- An exited process whose parent did not `wait()` on the child
- Does not consume resources beyond a PID and meta information storage (< 1k generally)
- Generally caused by two situations:
  - Bug in software
  - Overly taxed machine

SIGNALS

- First form of Interprocess Communication (IPC)
- A signal is a message sent to a process to indicate events or other conditions. The signal itself is the message - there around three dozen defined signals...

COMMON SIGNALS

- **HUP** - *Hangup*
- **INT** - *Interrupt*
- **QUIT** - *Quit*
- **ILL** - *Illegal Instruction*
- **ABRT** - *Abort*
- **KILL** - *Kill*
- **SEGV** - *Segmentation Fault*
- **ALRM** - *Alarm*
- **TERM** - *Terminate*
- **STOP** - *Stop*
- **CONT** - *Continue*
- **FPE** - *Floating Point Exception*

SENDING SIGNALS

- `kill`: Send a signal to a process. Default signal: TERM
- Examples:
  - `kill 457`
  - `kill -9 2359`
  - `kill -CONT 1350`

```
slideshow.end();
```