Processes and Process Creation

UNX511 Week 6 Class 1

John Sellens

September 5, 2025

Seneca Polytechnic

Outline

Processes – Let’s Review

Processes

- Programs (commands) run in processes
- All processes are children of some parent process
- Processes are a tree – all descendants of process ID 1
- Processes have memory, and use resources
- Processes have attributes – process ID, user, working directory, umask, etc.
- Shells run commands for us
  - By creating processes
  - Standalone, or in pipes
  - And providing job control tools (fg, bg, kill, etc.)

Process Memory Usage

- A process's memory is organized into "segments"
  - Text, data, heap, stack, ...
- See the discussion `https://github-pages.senecapolytechnic.ca/unx511/Week6/ProcessAddressSpace.docx`

Processes and Process Creation

Process Creation

- Every process is (was) created by a parent process
- The parent calls `fork(2)` (or `vfork(2)`) to create a copy of itself – the child process
  - The child process gets a copy of the parent's memory
- And then the new child process either continues running the same program,
- Or replaces itself with some other program using `execve(2)` or `exec(3)`

System Calls and Library Functions

- `fork(2)` – create a new process, with copy of memory
- `vfork(2)` – create a new process, but delay copying memory
  - More efficient if you're going to immediately `execve()`
- `execve(2)` – replace running program with a different program
- `exec(3)` – handy cover functions for `execve()`
- `wait(2)` – wait for a (any) child process to exit
- `waitpid(2)` – wait for a particular child process to exit
- See a summary of these functions in `https://github-pages.senecapolytechnic.ca/unx511/Week6/ProcessCreationAndTermination.docx`

System Calls and Library Functions, cont'd

- `_exit(2)` – terminate calling process immediately
- `exit(3)` – tidy up and then call `_exit()`
- `on_exit(3)` – call a function on `exit()` – Linux specific
- `atexit(3)` – call a function on `exit()` – POSTX

Process Code Samples

- 1_fork – create a child, continue same program
- 2_exec – create a child, run a different command
  - The shell does this *very* often
- 3_on_exit – call function(s) on exit
- 4_sysmonFork – monitor network interfaces with child processes
- 5_sysmonExec – the same, but run a separate monitor command
- Let's have a look ...

Summary

- As always, processes are a fundamental aspect of Linux/UNIX systems
- System programmers (and most others) need to have an understanding of process creation and management
- Next class: signals!