Operating Systems - Lab 4

Wednesday 15th March, 2023

Instructions

- Group enrolment on Themis
  https://themis.housing.rug.nl/course/2022-2023/os
- Submit in pairs
- Deadline: 31st March, at 23:59
- Programming language: C

Choices

You have the choice between two assignments for the final lab:

- Shell part 3
- Userland exec

You have 2 different ways to get a bonus:

- Up to +2: Implementing the bonus of the chosen assignment.
- +4: Implementing both assignments.
  - You do not need to do any bonuses for the 2 assignments to get the +4 if you do both.
  - You cannot get more than +4 on this assignments.
  - Basically, if you are doing both assignments, you don't need to do the bonuses (unless you really want to).

Shell - Requirements

- Lab 1:
  - Execute a program from user's search path ($PATH).
  - Command composition (&&, ||, ;)
  - String parsing
- Lab 3:
  - Pipeline
  - I/O redirection
  - cd builtin
- Lab 4:
  - Background processes
  - jobs
  - kill

Shell - Background processes

- & operator used for defining background processes.
  - `echo "a" \& echo "b" \& echo "c" \&`
    - Prints "a", "b" and "c" in an undefined order
  - `sleep 10 & echo "a"`
    - Will print "a" immediately.
- Relatively easy to implement using the "fork - exec" model.
  - Simply do not wait on the child to finish.
- Program must not stop when exit is called while background processes are active.
  - How to test: "`sleep 1 & exit`" should not exit the shell.
  - CTRL + C should produce the exact same behaviour as exit
    - CTRL + C behaviour can be modified by using a signal handler for SIGINT.

Shell - jobs

- The built-in command "jobs" should list the active background processes.
  - You need to keep track of when background processes terminate.
    - You can do this through a signal handler. Listening for SIGCHLD allows you to track terminated child processes.
    - You still need to use wait to clean up children, even if listening to SIGCHLD.
- No jobs available:
  - No background processes!
- Jobs available:
  - Process running with index 3
    Process running with index 2
    Process running with index 1

Shell - kill

- The built-in command "kill" should allow a user to kill an active background task.
  - PIDs are hard for users to keep track of. Users should kill processes through the index provided in "jobs".
    - Up to you to translate from index to PID.
  - User can provide a signal to send to the background process.
  - Default signal is SIGTERM.

Shell - Signal handler

- Implemented through sigaction().
- Install new signal handler through
  - `sigaction(signum, sigaction, NULL);`
    - signum is the signal to be caught by the handler.
    - sigaction is of type "struct sigaction". It is up to you to read the documentation for this struct. This struct implements your new signal handler.

Userland exec

- Implement the exec function in userland. This means the kernel does not handle the system call.
- Will teach you about ELF files, memory management, and the program stack.
- The process:
  1. Clear all memory of the calling process.
  2. Load the ELF file binary into memory.
  3. Set up a stack for the new binary.
  4. Run the new binary.
- Program is statically linked. Dynamic linking is a bonus.
- This assignment's challenge lies in technically understanding the process. Please read the full PDF before starting or asking questions.