CS312 Lab 3

August 21, 2024

- Part A has to be shown in the lab on the same day and has to be submitted on Moodle before midnight the same day (applies to only Q4 in this lab.) A plagiarism check may be done on the submissions.
- Submit as a single archive `lab3_rollnum.tgz.` This should contain a directory `lab3_rollnum/` with a single C source file named `lab3Q4_rollnum.c.`
- Part B is take-home. This will not be evaluated. You do not have to submit on Moodle.

Part A

1. Run the programs from Chap 5 of the text. These are provided as `remziCpu.tar.gz`. Understand how they work.
2. Understand the commands `ps` and `top`. Check out their common options.
3. Understand the `/proc` filesystem. Read the man page (`man proc`).
4. Write a `HelloWorld` program with an infinite loop.
   (a) Print its process id (pid).
   (b) While the program is running, use the `ps` command with the correct options to determine the pid, parent's pid. How much memory is being used by the program? Use the RSS field to determine this.
   (c) Modify the code to initialize a large (at least 10^6 elements) array of type double. Does the memory usage of the program change?
   (d) Look up the program details in the `/proc` directory. Check the `fd` directory.
   (e) Through the shell, redirect the output of the program to a local file. Run the program and check the `fd` directory again. What do you observe?

Part B

5. Create a mini version of the `ls` program called `minils`. Use the library functions `opendir()`, `readdir()` and `closedir()`.
   (a) Make the basic version first. Verify that the output is correct with the `ls` command.
   (b) Handle the option `-S`. See the `ls` man page to see what it does. Can you implement other common options, such as `-l`?
   (c) Handle errors "Permission denied", "Directory does not exist".

(d) Can the standard Bash shell perform piping and redirection on your `minils` program?

(e) Add `minils` to the current Bash path.

6. Create a simple shell. A code snippet `shell.c` is given, that tokenizes the input. Your shell must be able to run any Linux command available on your system. You must do this using `fork`, `exec` and `wait` system calls. Do not use the `system` library function. Aside: whats the difference in both these ways? The man page of the `system` library call should help.

   The shell should have the following features:

   (a) It should have the prompt %.

   (b) Ctrl+C will terminate the shell. Otherwise it should keep running.

   (c) The user will enter standard Linux commands with supported options and your shell will execute the command. The output will be displayed (including any errors made by wrong options etc.) After that the shell prompt will return, waiting for the next command. Verify the output using a standard Bash shell.

   (d) The shell must handle basic errors including "command not found". Just pressing enter will simply return the prompt.

   (e) Once the basic shell is ready, implement the `cd` command, using the `chdir` system call. Verify with the `pwd` command.

   (f) Run the `minils` program through your shell.

   (g) Make sure that all child processes are correctly reaped by the shell when they exit. Verify with `ps`.

   (h) Think about how to implement more advanced features including background execution, command redirection.

Acknowledgments: This question and the skeleton code is derived from Prof. Mythili Vutukuru's Operating Systems course at IIT Bombay.