Haley Burnell

Dr. Raj

CSC 4100

25 February 2024

**ExploreCSC4100/5100 – Homework 1 – Processes**

**https://pages.cs.wisc.edu/~remzi/OSTEP/cpu-api.pdf**

1. Write a program that calls fork(). Before calling fork(), have the main process access a variable (e.g., x) and set its value to something (e.g., 100). What value is the variable in the child process? What happens to the variable when both the child and parent change the value of x?

**A screen shot of a computer

Description automatically generated**

As seen above, the parent seems to execute and end and then the child gets called and they seem to have their own independent copies of the variable x=100 they share.

1. Write another program using fork(). The child process should print “hello”; the parent process should print “goodbye”. You should try to ensure that the child process always prints first; can you do this without calling wait() in the parent?

**A screenshot of a computer program

Description automatically generated**

As seen above, when the child spawns, it has its own address space, registers and more so it’s return value is different. It turns out the CPU Scheduler has freedom of when to handle these two processes once the child is spawned off which results in **non-determinism** and potential problems with multi-threaded programs.

1. Write a program that calls fork() and then calls some form of exec() to run the program /bin/ls. See if you can try all of the variants of exec(), including (on Linux) execl(), execle(), execlp(), execv(), execvp(), and execvpe(). Why do you think there are so many variants of the same basic call?

**A computer screen shot of a program

Description automatically generated**

Just from their parameters, it seems like some of the functions take a pathname and others take a file:

**A screenshot of a computer

Description automatically generated**

* the v family (execv, execvp, and execvpe) take an argument that is an array of pointers available to the new program
* the e family (execle, execvpe) have an environment argument
* the p family(execlp, execvp, and execvpe search for an executable file and search the PATH variable

(see exercise\_3\_fork\_extended.c for the implementations of all six options). I think there are many variants because some are more friendly to parallelization than others.

1. Write a program that creates two children, and connects the standard output of one to the standard input of the other, using the pipe() system call.

* See exercise\_4\_fork.c