CECS 326-01 Operating Systems

William Luu

Assignment 1

Due Date: 10/01/2024

Submission Date: 10/01/2024

Program Description

The program consists of two components: parent.cpp and child.cpp. The parent program is designed to take multiple name and age pairs as command-line arguments. It will fork a new process for each pair and pass the relevant data (name and age) to the child program (child.cpp). Each child process will print the name and age in a specific format. After all child processes finish execution, the parent process will summarize the number of child processes and terminate.

child.cpp

child.cpp is a simple program that utilizes command line arguments, with “int argc” indicating the amount of arguments used in the command line. “char *argv[]” is an array of character pointers which each element points to a string representing a command-line argument. For example argv[0] would be the name of the program. In main, child.cpp outputs “I am " << argv[1] << ", " << argv[2] << " years old" which displays the child’s name and age.

parent.cpp

parent.cpp initializes a vector “childPIDs” that holds process IDS of the children process. Main will loop i from range 0 to “argc”, incrementing twice for each child. Every incrementation, “pid_t pid = fork()” is called which returns -1 for error, 0 for the child process, and greater than 0 for the parent process which pushes process IDs of the childs into “childPIDs” to later be used to wait for each child process to finish running before having the parent process run. When “pid == 0” I set up an argument list for the execvp(), where args[] contains the name of the program to be executed, command-line arguments relevant to the child’s name, age pairs, and a NULL to indicate the end of the argument list. “execvp(args[0], args);” is later called to replace the current function with the child executable aka “./child”. When that is done, loop through each process ID from the vector holding the process IDs and wait for each child process to finish running. Finally run the parent process and return 0.