CSCI4730/6730 – Operating Systems

Project #1: Multi-process and IPC

Due date: 11:59pm, 02/03/2022

Description

In this project, you will design and implement a multi-process word counting program. The single-process version[1] is provided and you will convert it into the multi-process architectures.

Multi-process Word Counting Program (50%)

The main problem of a single-process program is the scalability. It cannot scale large number of words.

To address the problem, you will convert the word counting program into the multi-process model. The main process creates multiple child processes and effectively divides work between child processes. The child process sends the result to the main process via Inter-process communication channel. We will use **pipe** in this project. The main process waits children processes and reads the result via IPC channel, and prints out the total on the screen.

- You will modify "wc.c" to a multi-process model.
- The program receives the number of child processes and an input file name through the command-line argument.
    - 1st argument: target file
    - 2nd argument: # of child processes (default is 0)
    - 3rd argument: crash rate (default is 0%)
    - Example: *./wc ./large.txt 4*
- Explain your program structure and IPC in README.pdf file. **Only "pdf" format** will be accepted.

[1] It is slightly modified from the code in http://www.opentechguides.com/how-to/article/c/72/c-file-counts.html

Crash-handling (50%)

If one or more child processes crash before they complete the job (before sending the result through pipe), the final output will be incorrect. In this project, we will monitor the exit status of each child. If there is one or more child processes crashed, we will create new child processes to complete the job.

The parent process should "wait" for all child processes and monitor their exit status (hint: use waitpid()). If an exit status of a child process is "abnormal termination" by a signal, the parent process creates a new child process to re-do the incomplete job.

- You can use 3rd command-line arguments (integer between 1 and 50) to trigger crash. 50 means each child process has 50% chance to be killed abnormally by signal. 0 means 0% chance to crash.
- Explain how your program handles crash in README.pdf file.

Submission

Submit a tarball file using the following command

%tar czvf p1.tar.gz README.pdf Makefile wc_multi.c

1. README.pdf file with:
    a. Your name
    b. Explain your design of multi-process structure and IPC.
    c. Explain how your program handles crash.
2. Submit a tarball through ELC.