CS370 – Homework 3

Pipes and Shared Memory

Program Description

- **Initiator** receives the filename through the command line argument.
- **Initiator** then creates a pipe and checks for successful creation.
- Pass the pipe reference to **Reader** for maintaining a running sum of the inputs.
- **Reader** writes the sum to the pipe using the provided reference. (only the write end is required)

Program Description

- The control is passed back to the **Initiator** file where we reads contents of the pipe into a char array.
- **Initiator** creates three shared memory segments, for **Pell**, **Composite**, and **Total**.
- Further, we print the name and the file descriptor of the shared memory.
- Fork the **Pell**, **Composite**, and **Total** programs, and pass the name of the corresponding shared memory segment.

Program Description

- The **Pell**, **Composite**, and **Total** write the last value calculated to the shared memory segment.
- **The three child processes must run concurrently and not sequentially.**
- **Initiator** waits for all the child processes to complete and then prints the return value from the shared memory.
- Finally, unlink the shared memory.

Run Processes Concurrently

- In Assignment 2, the wait condition for the child was written before the parent process forked the next child.
- This leads to linear/sequential execution. However, for this Assignment, we need to execute the programs concurrently.
- Hence, the **Initiator** must fork all the child processes and then use the wait() command for each of them.

Function Description

• pipe() • shm_open() • ftruncate()	• mmap() • shm_unlink() • sprintf()

pipe()

Syntax: int pipe(int pipefd[2]);

Arguments: **pipefd**[2] is the array to represent two ends of the pipe. Each end is a file descriptor (FD).

Example: int pipefds[2];

int result_pipe = pipe(pipefds);

shm_open()

Syntax: int shm_open(const char *name, int oflag, mode_t mode);

Arguments: **name**: name of the memory segment

oflag: can take the following values: O_RDONLY, O_RDWR, O_CREAT, O_EXCL, O_TRUNC

mode: permissions in the form 0666

Example: char shm_Name[15] = “Shared_Mem0”;

int shm_fd = shm_open(shm_Name, O_CREAT | O_RDWR, 0666);

ftruncate()

Syntax: int ftruncate(int fd, off_t length);

Arguments: **fd:** is the file descriptor

length: is the desired size of the memory segment. (Will be initialized to 0)

Example: int result = ftruncate(fd, 1234);

mmap()

Syntax: void *mmap(void *addr, size_t length, int prot, int flags, int fd, off_t offset);

Arguments: **addr**: beginning address of the memory object

length: length of the memory object in bytes

prot: protection of the pages (PROT_EXEC, PROT_READ, PROT_WRITE, PROT_NONE)

flags: Updates to the mapping should be visible to other processes mapping the same region. (MAP_SHARED, MAP_PRIVATE etc.)

mmap()

Arguments: **fd:** returned by shm_open

offset: is 0 in here

Example: mmap(0, size, PROT_READ, MAP_SHARED, shm_fd, 0);

shm_unlink()

Syntax: int shm_unlink(const char *name);

Arguments: **name:** is the memory object name to be unlinked

Example: int error = shm_unlink(shm_Name);

sprintf()

Syntax: int sprintf(char * buffer, const char * string, ...);

Arguments: **string** is stored in **buffer**

Example: sprintf(buffer, "Sum = %d", sum);

Makefile

- Following change is needed in your Makefile from Assignment 2
  - Add **–lrt** during compilation to call `shm_open()` and `shm_unlink()`

    (see point 4 from Notes in last page of Assignment 3)

Other Requirements

- Code should compile and run on CS Department computers.
- Submit all .c, along with Makefile and README.txt

Quick Review

- **Initiator** calls the **Reader** by passing it the file name and the write end file descriptor of the pipe. The **Reader** maintains a running sum of each line and writes the sum to the pipe.
- **Initiator** forks three child processes **Pell**, **Composite**, and **Total**. Three shared memories are created for each child to write the last calculated result.
- Remember to **unlink** the shared memory!

Project Grade and Deadlines

Deliverable	Date	Points
D1 Team composition and idea proposal	10/1/2021	5%
D2 Progress report	11/4/2021	15%
D3 Slides and final reports	12/2/2021	75%
D4 Presentations/Demos	12/6/21 or 12/8/21	
D5: Peer Reviews	12/11/2021	5%

This assignment would be worth **10** points towards your final grade. D3 and D4 combined are worth 75%. This includes your participation in viewing the work of other fellow students and participation in the related events.

Option A: Research a technical topic

- A list of topics is available
- Must be of current interest and related to operating systems
- You will research using in-depth research/industrial publications and news articles.
- You will learn about the state of the art and current technological trends.
- A formal report is the specified format will be needed. A presentation using slides/poster is also required.

Option B: An embedded/IoT system

- Project must involve a platform (such as a Raspberry Pi) with WiFi capability which will communicate with at least **one other computer** and with **at least one sensor**.
- You will specify the project objective, select the appropriate platform, hardware/software needed. Your proposal must include a justification of the choices.
- After you have built your system, you must evaluate your project for at least two of the following attributes
  - Potential security holes
  - Power consumption
  - Limitations like resolution or accuracy etc., (Cost and marketability of the project)
- A report, a demonstration, and a set of slides will be needed.
- How to set up your Raspberry Pi. You need to locate similar information for other boards.

Project Choices

- Teams: Team sizes of 2 or 3. You get to choose! Work should be divided evenly among group members.
  - Choice A: Research a technical topic of current interest. A list of topics is already available. The topic needs to be pre-approved.
  - Choice B: An embedded/IoT development project. Research the aim, platform, obtain hardware/software. Need a critical design evaluation and a demo.

Acknowledgements

- These slides are based on contributions of current and past CS370 instructors and TAs, including J. Applin, L. Mendis, M. Warushavithana, S. R. Chowdhury, A. Yeluri, K. Bruhwiler, Y. K. Malaiya and S. Pallickara.

Thank You

Questions?