CS370 - Homework 3

Pipes and Shared Memory

Program Description

- **Generator** receives the filename through the command line argument.
- **Generator** 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 **Generator** file where it reads contents of the pipe into a char array.
- **Generator** finds the sum of all digits from an integer value of the char array.
- **Generator** creates three shared memory segments, for **Prime**, **Square**, **Composite**.
    - ✓ **Composite** and **Perrin** will share a memory segment.
- Further, we print the name and the file descriptor of the shared memory.

Program Description

- Fork the **Prime**, **Square**, and **Composite** programs, passing the name of the corresponding shared memory segment as an argument.
- **Composite** will spawn a **Perrin** process using the same process as **Initiator.**
- The **Prime**, **Square**, **Composite**, and **Perrin** will write the last value calculated to their respective shared memory segment.
- **Three child processes, Prime, Square, Composite must run concurrently, and Perrin runs sequentially from Composite.**
- **Generator** waits for all the child processes to complete and then prints the return value from the shared memory.
- Finally, unlink the shared memory.

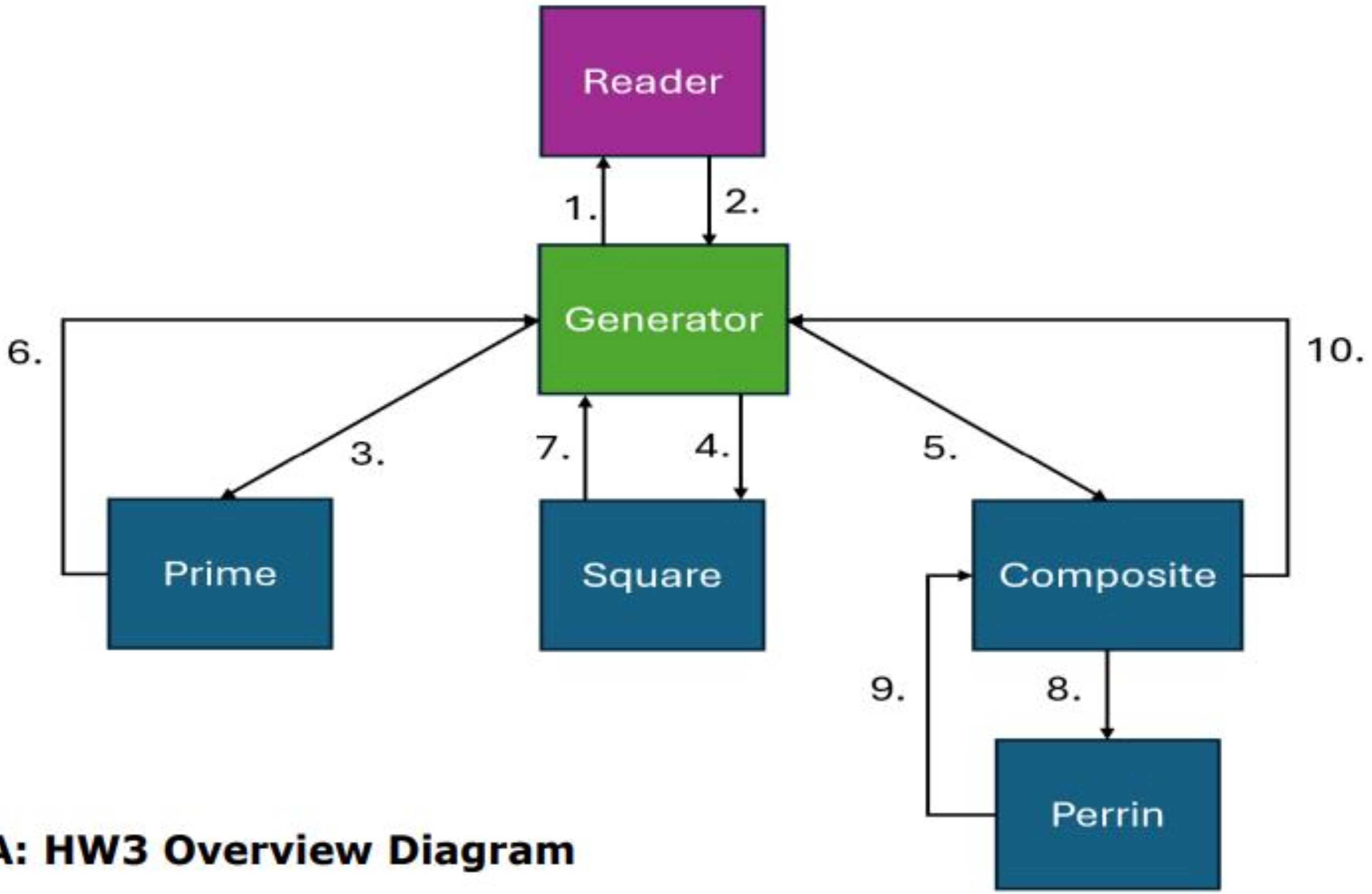


Fig. A: HW3 Overview Diagram

Program Overview

1. Generator spawns (fork/exec) a Reader process passing the input filename.

2. Reader opens file handle and processes input. Returns an integer via a shared pipe. Generator reads from the pipe and processes that input to create some number N.

3. 4. and 5. Generator creates shared memory and spawns Prime, Square, and Composite processes, passing them N and the shared memory address for their pairwise communication.

6. and 7. Prime and Square will pass the results of their calculation directly back to generator which will handle them as directed.

Program Overview

8. In addition to its regular sequence computation, Composite must create a shared memory segment and spawn a Perrin process passing it N and the address of that segment.

9. Once Perrin has performed its computations, it will write to shared memory and Composite will read that result.

10. Composite will write it's final value and Perrin’s final value to 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 and sequentially.
- Hence, for the concurrent processes the **Generator** must fork three child processes and then use the wait() command for each of those.

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

- Makefile provided, please use it. This is the file we will use to test your program. So, its best if you use it while completing the assignment.

Other Requirements

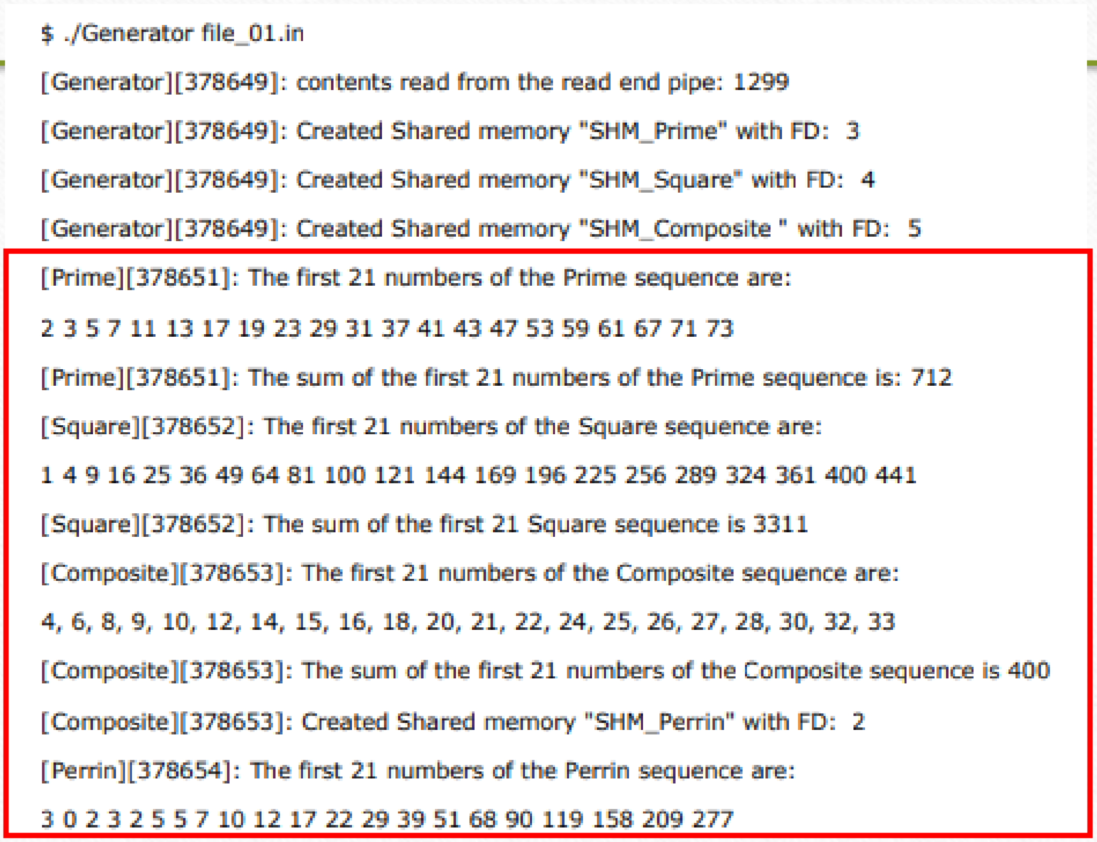
- Code should compile and run on CS Department computers.
- Submit all .c, along with Makefile and README.txt. **Please remember to submit your assignment in a zipped file.**

Resources

- Read & Write with Pipe
- POSIX Shared Memory

Demo of Concurrent Program

The order of print statements can be varied at the time of your run



```
$ ./Generator file_01.in
[Generator][378649]: contents read from the read end pipe: 1299
[Generator][378649]: Created Shared memory "SHM_Prime" with FD:  3
[Generator][378649]: Created Shared memory "SHM_Square" with FD:  4
[Generator][378649]: Created Shared memory "SHM_Composite " with FD:  5
[Prime][378651]: The first 21 numbers of the Prime sequence are:
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73
[Prime][378651]: The sum of the first 21 numbers of the Prime sequence is: 712
[Square][378652]: The first 21 numbers of the Square sequence are:
1 4 9 16 25 36 49 64 81 100 121 144 169 196 225 256 289 324 361 400 441
[Square][378652]: The sum of the first 21 Square sequence is 3311
[Composite][378653]: The first 21 numbers of the Composite sequence are:
4, 6, 8, 9, 10, 12, 14, 15, 16, 18, 20, 21, 22, 24, 25, 26, 27, 28, 30, 32, 33
[Composite][378653]: The sum of the first 21 numbers of the Composite sequence is 400
[Composite][378653]: Created Shared memory "SHM_Perrin" with FD:  2
[Perrin][378654]: The first 21 numbers of the Perrin sequence are:
3 0 2 3 2 5 5 7 10 12 17 22 29 39 51 68 90 119 158 209 277
```

Thank You

Questions?

Acknowledgements

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