CSE344 - System Programming - Homework #1

Fatih Kaan Salgır - 171044009

- Maximum number of fifo to handle is defined as N

1 Problem Solution Approach

1. First process will create the shared memory, and all of them are going open it.
2. The process will create the fifo. This fifo will be the fifo that the process reads. If the process has a potato, it will register to shared memory with the pid. And wait for creation of all fifos.
3. All of the fifos are going to be opened. 1 one of them to read, other ones to write.
4. If process has a potato it will write its pid to a random fifo.
5. The process will read its fifo until it reads either a potato id or a termination message (integer, defined as -2) when all potatoes are cooled down.
6. It will gets the potato with potato id, increments its number of switch, checks if cooled down. If its the case, it will let others know and terminate. Otherwise it will write again to some random fifo.

1.1 Shared Memory Initilization

First `shm_open` is called with both `O_CREAT` and `O_EXCL` flags. As the man page of the `shm_open` states, if the shared memory already exist, it will return -1 and `errno` will set to `EEXIST`. If shared memory already exists then, simply `mmap` can be used to establist the mapping address. If shared memory is created, some initialization steps needed:

- `ftruncate` will be called to set its size.
- Semaphore in the shared memory will be set to 0.
- number of created files will be set to 0.
- number of potatoes will be set to 0.

1.2 Shared Memory Design

`struct shared_memory` is used in order to organize shared memory.

all fifos are created	[sem _t]
number of created fifos	[int]
size of potatoes	[int]
#1 potato	[struct{int, int, int}]
#1 id	[int]
#1 switch count	[int]
#1 number of switches	[int]
#2 potato	[struct{int, int, int}]
#2 id	[int]
#2 switch count	[int]
#2 number of switches	[int]
...	

1.3 Waiting for FIFO Creation

1. A process is going to create their fifo and wait for other process using `sem_wait()`. This semaphore is different from the one given as command-line arguement. It lies in the shared memory, and inital value is 0. (`all_fifos_created`)
2. Processes will create the fifos one by one, to avoid race condition. The synchronization technique is used here is named semaphore.
3. If a process has been reached to `EOF`, so this process is the last one. This means all other processes are ready to continue. It will increment to value of `sem_post()` N times (number of processes).