Gabriel Hofer
April 11, 2021
Operating Systems
Exam #2

How to Compile and use the program

Compile the program by entering 'make' in the terminal. This will make the executable 'dash'. Then enter the command: './dash' which will run the program executable.

This program implements redirected input and output and also pipes. This program also implements a PID Manager and a test program. I used code from the following links to implement pipes and I/O redirection:

- https://www.cse.sdsmt.edu/ckarlsso/csc458/spring21/src/ALP/jchen.c
- https://www.cse.sdsmt.edu/ckarlsso/csc458/spring21/src/ALP/pipe1.c

I/O Redirection & Pipes

The example code provided by Dr. Karlsson was very helpful for this part of the assignment. For implementing pipes, one modification that was made to Dr. Karlsson's code was changing the following lines:

```
execl("/bin/cat", "cat", "pipe.txt", 0);     -->     execl("/bin/sh", "sh ", "-c", a, NULL);
execl("/bin/sort", "sort", 0);               -->     execl("/bin/sh", "sh ", "-c", b, NULL);
```

This caused execl to run commands in the strings a and b instead of calling sort and cat. Similarly, the code in https://www.cse.sdsmt.edu/ckarlsso/csc458/spring21/src/ALP/jchen.c was modified by changing argv[2] and argv[3] to strings a and b, respectively so that the input was redirected to the filenames contained in a and b instead of command lines arguments.

```
Argv[2] --> a
Argv[3] --> b
```

PID Manager

- int **allocate_map**(void) -- Creates and initializes a data structure for representing pids; returns -- 1 if unsuccessful, 0 if successful
- int **allocate_pid**(void) -- Allocates and returns a pid; returns -- 1 if unable to allocate a pid (all pids are in use)
- void **release_pid**(int pid) -- Releases a pid

Instead of creating threads, the testpid function forks 10 times in a loop and each child process represents a new "thread". Then each thread calls the sleep function for a random amount of time. When sleep returns, release_pid is called by the thread.

Memory Manager

The memory manager program accepts one integer argument. Since the page size is 4KB, we find the page number by dividing the integer argument by 2^12. Then we find the offset on the page by modding the integer argument by 2^12:

Integer_argument % (2^12)