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 modified the 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 function `int handle_pipe(char * buf, int idx)` accepts two arguments: a character string `buf` containing the entire shell command and an integer `idx` pointing to the index of the ‘|’ pipe character in the character string (`buf`). `handle_pipe()` splits buf into two strings `a` and `b`. `a` contains the command before the ‘|’ (pipe) and `b` contains the command after the ‘|’ (pipe) character. The rest of the function calls `pipe()`, `execl()`, and `dup()` to execute the command in character string `a` and pipe the output to the command in character string `b`.

The example code provided in the writeup by Dr. Karlsson was very helpful for understanding how to do 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 made execl to run the commands stored in the strings `a` and `b` instead of calling `cat` and `sort`, respectively.

`int handle_gt(char * buf, int idx)` and `int handle_lt(char * buf, int idx)` handle I/O redirection. Similarly to `handle_pipe()`, these two redirection functions have the same function parameters. And, at the beginning of the function, they both split `buf` into two character strings `a` and `b`. `a` always contains the command string while `b` always contains the name of the file from which to read/write.

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:

argv[2] --> a
argv[3] --> b

Pipes were tested using the following commands (see if we can write some text to file):

/home/gdh/dash> echo "this is a test" | tee testfile
handle pipe
a: echo "this is a test"
b: tee testfile

this is a test
The child process id number is 10895 [10895]

/home/gdh/dash> cat testfile
Child PID: 10909
CPU time: 0.002623 sec user, 0.010536 sec system
this is a test

I/O redirection was tested with the following commands:

/home/gdh/dash> cat test>output3
handle_gt
strlen(buf): 17
a: cat test
b: output3
The child process id number is 11314

/home/gdh/dash> cat output3
Child PID: 11321
CPU time: 0.000000 sec user, 0.000000 sec system
this is a test

/home/gdh/dash> cat<letters
handle_lt
a: cat
b: letters
The child process id number is 12339
z
d
k
n
g
q
a

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.

PID manager was tested by entering the testpid command on the command line.

```
/home/gdh/dash> testpid
allocated 483
allocated 483
allocated 390
allocated 446
allocated 401
allocated 375
allocated 441
allocated 377
/home/gdh/dash> allocated 415
allocated 396
released 396
/home/gdh/dash> released 446
/home/gdh/dash> released 375
/home/gdh/dash> released 441
/home/gdh/dash> released 483
/home/gdh/dash> released 483
/home/gdh/dash> released 415
/home/gdh/dash> released 390
/home/gdh/dash> released 401
/home/gdh/dash> released 377
```

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)

The memory manager was tested by entering “memman 19986” on the command line in the dash shell:

```
/home/gdh/dash> memman 19986
The address 19986 contains :
page number = 4
offset = 3602
```

How to make and run

- Copy from submit directory to grading directory
- tar -xzf exam2.tgz
- cd exam2
- make
- ./dash
- Enter commands and compare output
- Assign a grade
- Rm -rf exam2