

# Operating Systems Lab

## Assignment 2

180010002

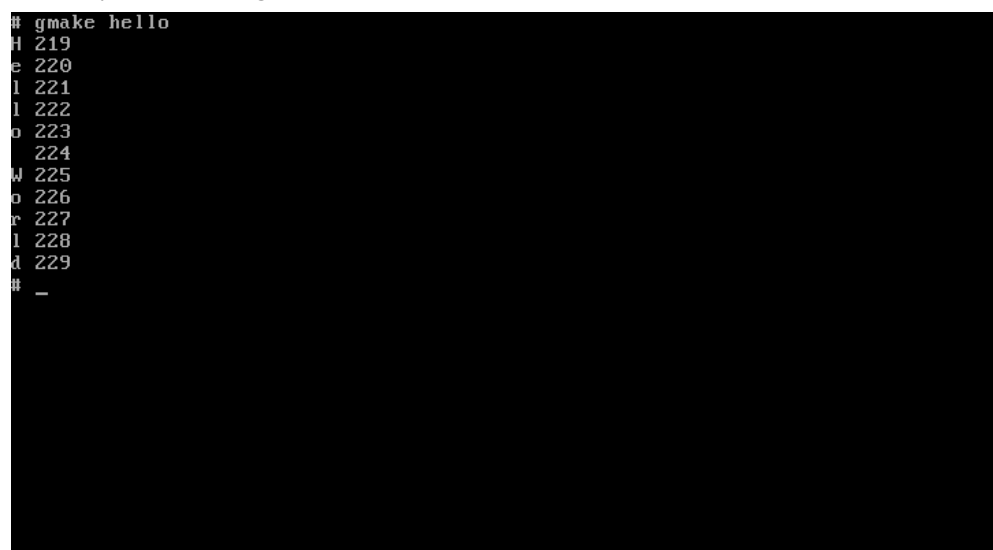
**Part I.** The goal here is to print “Hello World”, but each character is to be printed by different process and the process that prints the  $i$  th letter must have been spawned by the process that printed the  $(i-1)$  th letter.

To achieve this, we need to create nested processes, i.e., parent process creates child process and then child process creates grandchild process for the parent process and so on until each character is printed on the screen. So, I created a for loop of  $n$  times (where  $n$  is string length of “Hello World”), and before entering the loop created a child process. Within the loop, I checked whether the process is in child process, if yes, then printed the first character, sleep for 1 to 4 seconds then again created a child process within the child process thus creating the grandchild process. Then again check if the process is in child process or not, if yes then continue with the loop else wait for the child process to finish.

The minimum lines in which I could achieve this is 21 (including the lines to import libraries).

**Command: gmake hello**

The output is in the figure below:



```
# gmake hello
H 219
e 220
l 221
l 222
o 223
 224
W 225
o 226
r 227
l 228
d 229
# 230
_
```

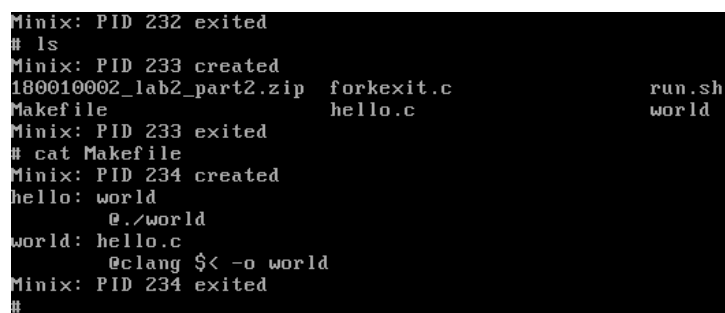
Figure 1. Part I expected output

**Part II.** The goal here is to modify the minix source code to print “Minix: PID <pid> created” everytime a new process is created and “Minix: PID <pid> exited” everytime a process ends.

To achieve this, the file *forkexit.c* located at *minix/servers/pm* is modified at appropriate place.

**Command:** `./run.sh`

Then after rebooting the minix system, we can see the print statements clearly on the screen whenever a process is created and ended. The corresponding screenshot is given below:



```
Minix: PID 232 exited
# ls
Minix: PID 233 created
180010002_lab2_part2.zip  forkexit.c      run.sh
Makefile                 hello.c         world
Minix: PID 233 exited
# cat Makefile
Minix: PID 234 created
hello: world
    @./world
world: hello.c
    @clang $< -o world
Minix: PID 234 exited
#
```

Figure 2. Part II expected output

The processes are created in tree like fashion where parent process is created first which creates a child process and processes exit in reverse order generally, child process exits first and then parent process exits. There could be cases where parent process exits unexpectedly, then the child process becomes orphan process and init process adopts it and ends it, since a child process cannot exist without its parent process.