OS Lab 2

Name : Hemanth Reddy
Roll : 180010023

Part-1

Implementation :

helloworld.c

Consider initially at i=0 the fork() will be called and since it is not equal to zero the program goes to else statement and makes the system call wait(NULL).
Since for a child the fork() value is 0 so the program goes through if statement and prints character 'H' with its pid and then it sleeps for some random time(using sleep() system call in rage (1-4)) and continues for loop, while its parent exits the program due to exit(0) system call.
Now the i value is 1 which again goes in the same way as above (previous child again forks another child) and this repeats till the last character of hello world.In the above process program that prints the i th letter have been spawned by the process that printed the (i-1) th letter.For each character the designated character is child pid printed below.All the conditions are satisfied as per question requirements.

```c
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<string.h>
#include<sys/wait.h>

int main(){
    char string[] = "Hello world";
    for(int i =0; i<strlen(string); ++i){
        if(fork() == 0){
            printf("%c : %d\n",string[i], (int) getpid());
            sleep(1+(rand() % 4));
        }
        else{
            wait(NULL);
            exit(0);
        }
    }
    return 0;
}
```

Minimum lines of C code with which you can achieve the above :

12 lines of code considering the code in the main function.

Makefile

gmake hello runs the below commands using the Makefile :

1. clang helloworld.c

   Compile c program using clang

2. ./a.out

   Run the executable

Screenshots :

Makefile is working and answer is printing correctly

```
  : 248
Minix: PID 249 created
w : 249
Minix: PID 250 created
o : 250
Minix: PID 251 created
r : 251
Minix: PID 252 created
l : 252
Minix: PID 253 created
d : 253
Minix: PID 253 exited
Minix: PID 252 exited
Minix: PID 251 exited
Minix: PID 250 exited
Minix: PID 249 exited
Minix: PID 248 exited
Minix: PID 247 exited
Minix: PID 246 exited
Minix: PID 245 exited
Minix: PID 244 exited
Minix: PID 243 exited
Minix: PID 242 exited
Minix: PID 238 exited
# _
```

Part-2

Implementation :

run.sh and forkexit.c

Run the required following commands using run.sh :

1) cp -f forkexit.c /usr/src/minix/servers/pm/forkexit.c

   Add these statements as per question requirements,

   a. printf("Minix: PID %d created\n", new_pid); in the [116] line of my forkexit.c code once the new_pid is created using new_pid = get_free_pid();

   b. printf("Minix: PID %d exited\n", mp->mp_pid); in the [245] line of my forkexit.c code before the process exits (exit_proc(..))

2) cd /usr/src/

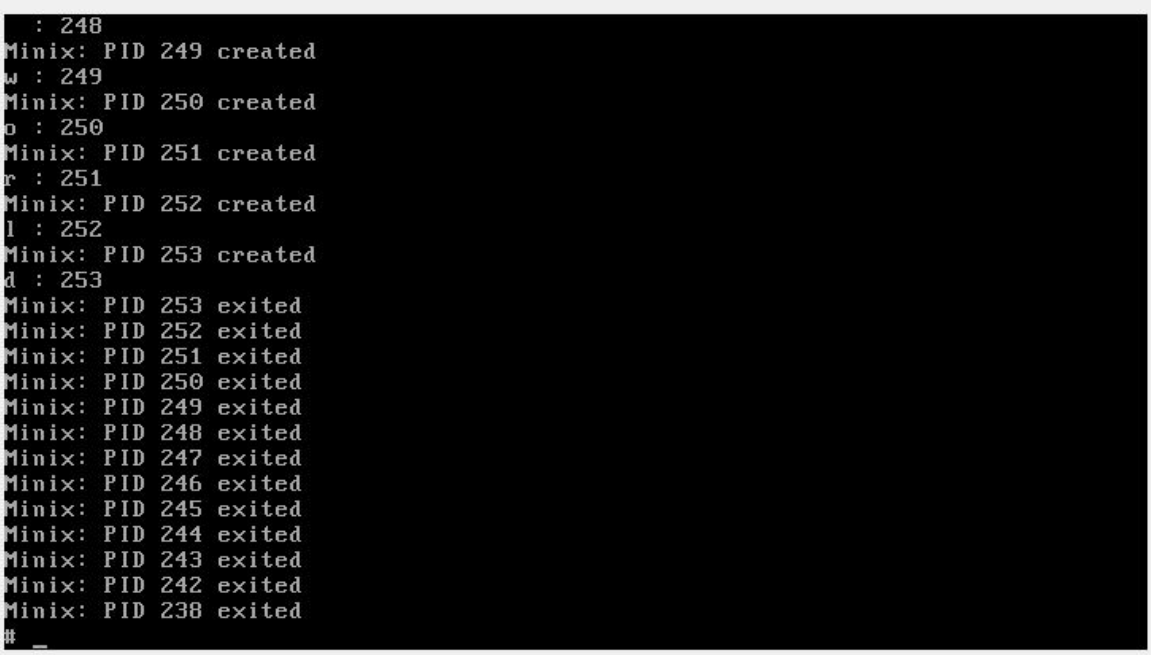
   a. Move to the minix folder(folder containing .git file) using above command

3) make build MKUPDATE=yes >log.txt 2>log.txt

   a. To see updated changes in your minix run the above make command.The log regarding the result is available in log.txt (gives error messages in case of any mistakes)

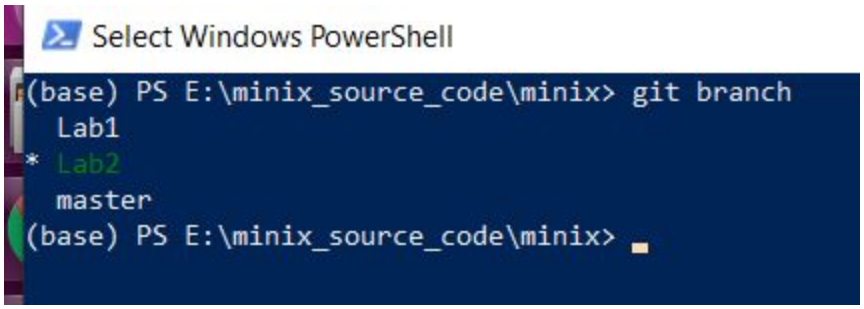
Comment on the order in which processes are created and processes exit and justify it is as expected

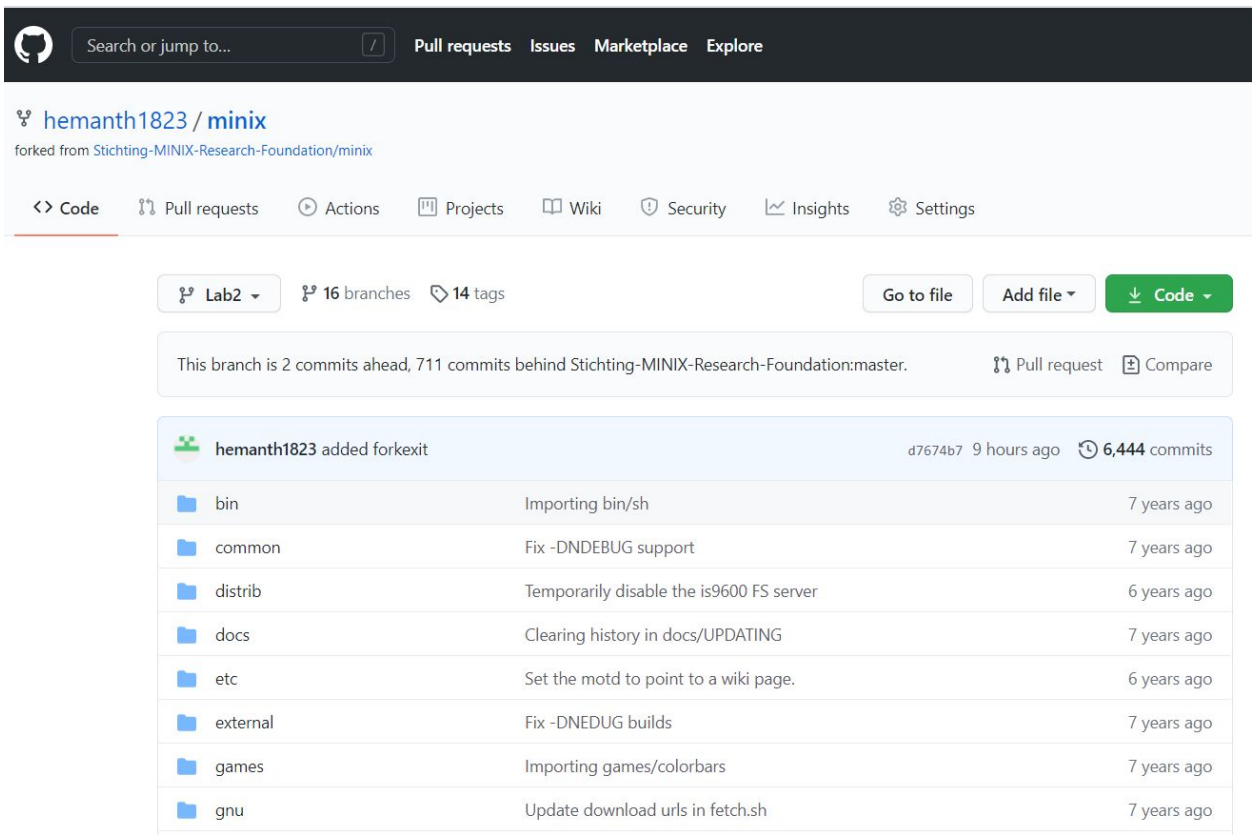
Consider initially at i=0 the fork() will be called and since it is not equal to zero the program goes to else statement and makes the system call wait(NULL).
Since for a child the fork() value is 0 so the program goes through if statement and prints character 'H' with its pid and then it sleeps for some random time(using sleep() system call in rage (1-4)) and continues for loop, while its parent exits the program due to exit(0) system call. Now the i value is 1 which again goes in the same way as above (previous child again forks another child) and this repeats till the last character of hello world.In the above process program that prints the i th letter have been spawned by the process that printed the (i-1) th letter.For each character the designated character is child pid printed below.
The Process that is printing 'H' has PID: 243. The process that prints the next character 'e' has PID:244 and so on... This tells that each process is a child of the previous process. It all gets terminated once in the end.(As seen in the end of second figure) like from d-253 to h-243.
All the conditions are satisfied as per question requirements that is as seen in the picture below the

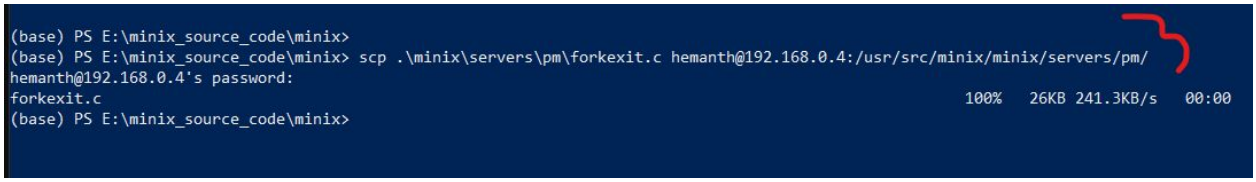


Screenshots :

Added new branch for this assignment







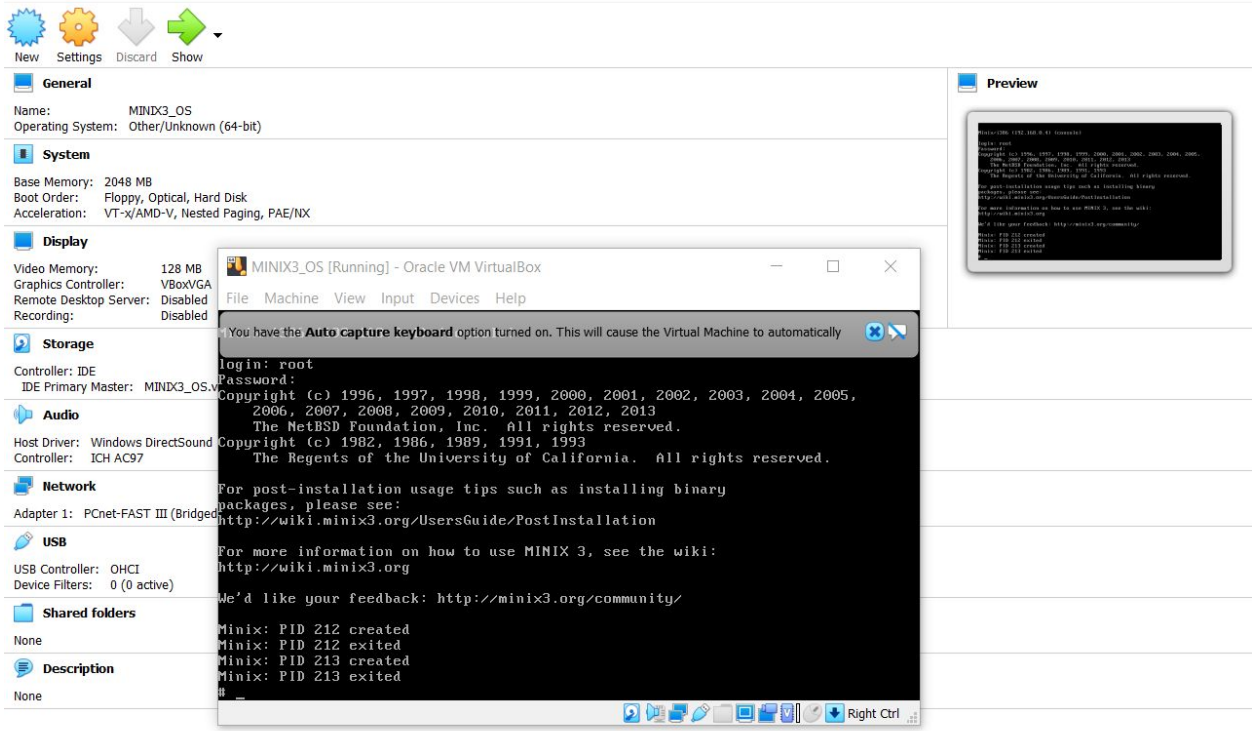
```
Makefile.inc drivers      lib          sbin         usr.bin
benchmarks   fs           llvm         servers      usr.sbin
bin          include      man          share
Minix: PID 230 exited
# cd servers/
# ls
Minix: PID 231 created
Makefile     devman       input        is           rs           vfs
Makefile.inc ds           ipc          pm           sched        vm
Minix: PID 231 exited
# cd pm/
# ls
Minix: PID 232 created
.depend    exec.d     glo.h      misc.d     proto.h    table.d    type.h
.gdbinit   exec.o     main.c     misc.o     schedule.c table.o    utility.c
Makefile   forkexit.c main.d     mproc.h    schedule.d time.c     utility.d
alarm.c    forkexit.d main.o     pm         schedule.o time.d     utility.o
alarm.d    forkexit.o mcontext.c pm.h       signal.c   time.o
alarm.o    getset.c   mcontext.d profile.c  signal.d   trace.c
const.h    getset.d   mcontext.o profile.d  signal.o   trace.d
exec.c     getset.o   misc.c     profile.o  table.c    trace.o
Minix: PID 232 exited
# vim forkexit.
forkexit.c forkexit.d forkexit.o
# vim forkexit.c
```

Showing that the changes in forkexit.c were made in minix system

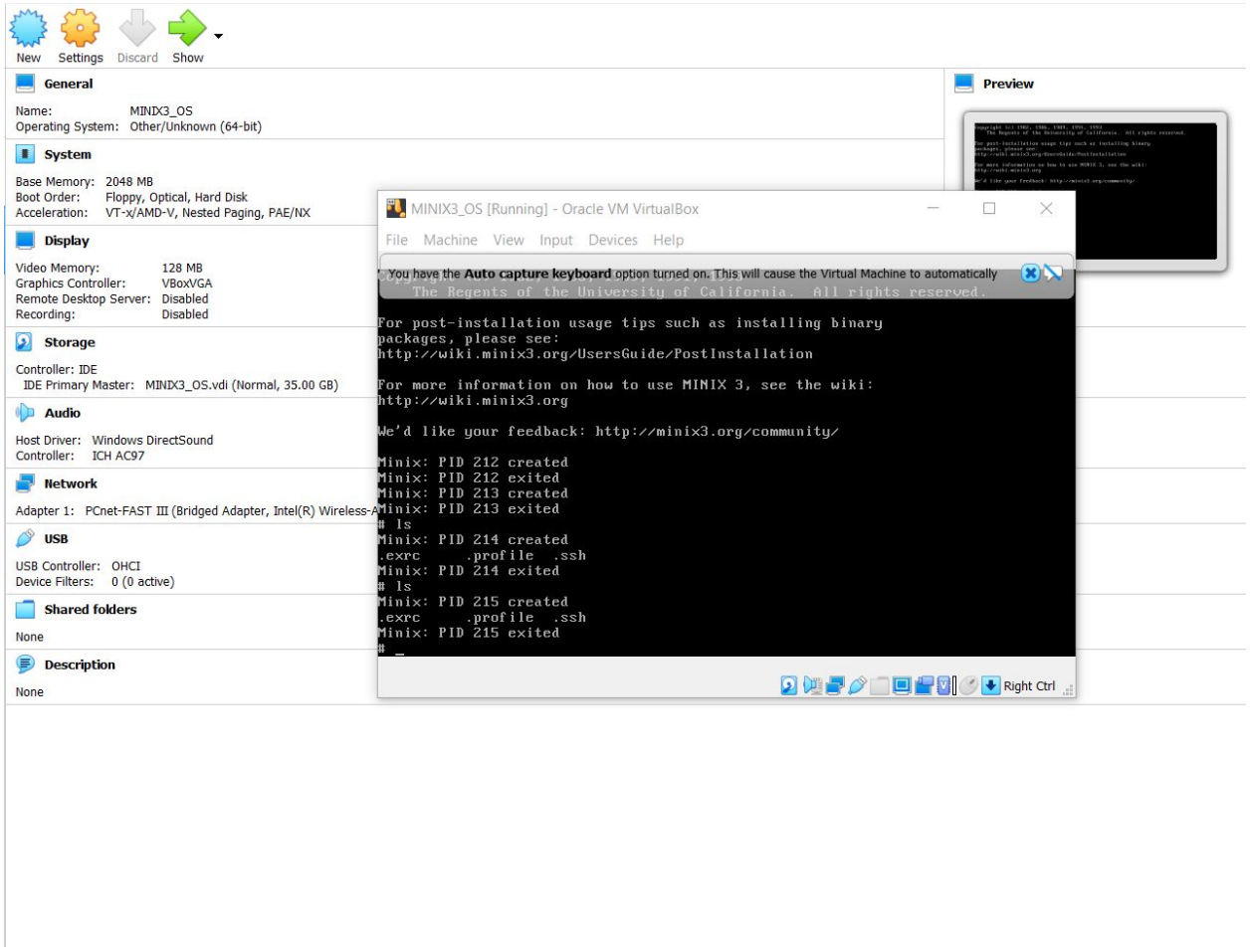
```
        rmc->mp_scheduler = SCHED_PROC_NR;^M
  }^M
^M
  /* Inherit only these flags. In normal fork(), PRIV_PROC is not inherited. */^
M
  rmc->mp_flags &= (IN_USE|DELAY_CALL|TAINTED);^M
  rmc->mp_child_utime = 0;              /* reset administration */^M
  rmc->mp_child_stime = 0;              /* reset administration */^M
  rmc->mp_exitstatus = 0;^M
  rmc->mp_sigstatus = 0;^M
  rmc->mp_endpoint = child_ep;          /* passed back by VM */^M
  for (i = 0; i < NR_ITIMERS; i++)^M
        rmc->mp_interval[i] = 0;        /* reset timer intervals */^M
^M
  /* Find a free pid for the child and put it in the table. */^M
  new_pid = get_free_pid();^M
  rmc->mp_pid = new_pid;        /* assign pid to child */^M
        ^M
  printf("Minix: PID %d created\n", new_pid);^M
  memset(&m, 0, sizeof(m));^M
  m.m_type = VFS_PM_FORK;^M
  m.VFS_PM_ENDPT = rmc->mp_endpoint;^M
  m.VFS_PM_PENDPT = rmp->mp_endpoint;^M
  m.VFS_PM_CPID = rmc->mp_pid;^M
```

```
int do_exit()^M
{^M
 /* Perform the exit(status) system call. The real work is done by exit_proc(),^
M
  * which is also called when a process is killed by a signal. System processes^
M
  * do not use PM's exit() to terminate. If they try to, we warn the user^M
  * and send a SIGKILL signal to the system process.^M
  */^M
  if(mp->mp_flags & PRIV_PROC) {^M
      printf("PM: system process %d (%s) tries to exit(), sending SIGKILL\n",^M
          mp->mp_endpoint, mp->mp_name);^M
      sys_kill(mp->mp_endpoint, SIGKILL);^M
  }^M
  else {^M
          printf("Minix: PID %d exited\n", mp->mp_pid);^M
      exit_proc(mp, m_in.m_lc_pm_exit.status, FALSE /*dump_core*/);^M
  }^M
  return(SUSPEND);              /* can't communicate from beyond the grave */^M
}^M
^M
/*===========================================================================*^M
 *                              exit_proc                                    *^M
@
```

Showing pid(process creating and exiting) once the minix machine started



Showing pid(process creating and exiting) for linux commands (shown ls in picture below)


New
Settings
Discard
Show
General
Name: MINIX3_OS
Operating System: Other/Unknown (64-bit)
System
Base Memory: 2048 MB
Boot Order: Floppy, Optical, Hard Disk
Acceleration: VT-x/AMD-V, Nested Paging, PAE/NX
Display
Video Memory: 128 MB
Graphics Controller: VBoxVGA
Remote Desktop Server: Disabled
Recording: Disabled
Storage
Controller: IDE
IDE Primary Master: MINIX3_OS.vdi (Normal, 35.00 GB)
Audio
Host Driver: Windows DirectSound
Controller: ICH AC97
Network
Adapter 1: PCnet-FAST III (Bridged Adapter, Intel(R) Wireless-A
USB
USB Controller: OHCI
Device Filters: 0 (0 active)
Shared folders
None
Description
None
Preview
MINIX3_OS [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
You have the Auto capture keyboard option turned on. This will cause the Virtual Machine to automatically
The Regents of the University of California. All rights reserved.
For post-installation usage tips such as installing binary
packages, please see:
http://wiki.minix3.org/UsersGuide/PostInstallation
For more information on how to use MINIX 3, see the wiki:
http://wiki.minix3.org
We'd like your feedback: http://minix3.org/community/
Minix: PID 212 created
Minix: PID 212 exited
Minix: PID 213 created
Minix: PID 213 exited
ls
Minix: PID 214 created
.exrc .profile .ssh
Minix: PID 214 exited
ls
Minix: PID 215 created
.exrc .profile .ssh
Minix: PID 215 exited
#
Right Ctrl