OS Hands-on Session

WEEK-2

Name: Vishwa Mehul Mehta

SRN: PES2UG20CS389

SEC: F

Program 1:

a) fork()

Fork system call is used for creating a new process (child process), which runs concurrently with the process that makes the fork() call (parent process).

```c
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
int main() {
    fork();
    printf("this is OS lab week2\n");
    return 0;
}
```

```
vishwa@Acer-Vishwa:/mnt/c/Users/Vishwa/Documents/Vishwa_PES/Sem4/OS/OS_HandsOn/week2$ ./fork
this is OS lab week2
this is OS lab week2
vishwa@Acer-Vishwa:/mnt/c/Users/Vishwa/Documents/Vishwa_PES/Sem4/OS/OS_HandsOn/week2$
```

b) getpid() and getppid()

getpid(): When any process is created, it has a unique id which is called its process id. This function returns the process id of the calling function.

getppid(): This function returns the process id of the parent function.

```c
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
int main() {
    pid_t process_id;
    pid_t parent_process_id;
    process_id = getpid();
    parent_process_id = getppid();
    printf("The process id: %d\n",process_id);
    printf("The process id of parent function: %d\n",parent_process_id);
    return 0;
}
```

```
vishwa@Acer-Vishwa:/mnt/c/Users/Vishwa/Documents/Vishwa_PES/Sem4/OS/OS_HandsOn/week2$ ./pid
The process id: 798
The process id of parent function: 9
vishwa@Acer-Vishwa:/mnt/c/Users/Vishwa/Documents/Vishwa_PES/Sem4/OS/OS_HandsOn/week2$
```

Program 2:

wait()

A call to wait() blocks the calling process until one of its child processes exits or a signal is received. After child process terminates, parent *continues* its execution after wait system call instruction.

```c
#include <stdio.h>
#include <sys/wait.h>
#include <unistd.h>
int main() {
    if (fork() == 0) {
        printf("this is a child process\n");
    }
    else {
        printf("this is the parent process\n");
        wait(NULL);
        printf("child process has terminated\n");
    }
    printf("end of program\n");
    return 0;
}
```

```
vishwa@Acer-Vishwa:/mnt/c/Users/Vishwa/Documents/Vishwa_PES/Sem4/OS/OS_HandsOn/week2$ ./wait
this is the parent process
this is a child process
end of program
child process has terminated
end of program
vishwa@Acer-Vishwa:/mnt/c/Users/Vishwa/Documents/Vishwa_PES/Sem4/OS/OS_HandsOn/week2$ Z
```

Program 3:

a) execvp()

Using this command, the created child process does not have to run the same program as the parent process does.

execvp.c:

```c
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
int main() {
    printf("this is execvp.c called by execvp() ");
    printf("\n");
    return 0;
}
```

execvp_main.c:

```c
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
#include <stdlib.h>
int main() {
    char *args[]={"./execvp",NULL};
    execvp(args[0],args);
    printf("end of program.\n");
    return 0;
}
```

```
vishwa@Acer-Vishwa:/mnt/c/Users/Vishwa/Documents/Vishwa_PES/Sem4/OS/OS_HandsOn/week2$ ./execvp_main
this is execvp.c called by execvp()
vishwa@Acer-Vishwa:/mnt/c/Users/Vishwa/Documents/Vishwa_PES/Sem4/OS/OS_HandsOn/week2$
```

b) execv()

This is very similar to execvp() function in terms of syntax as well.

execv.c:

```c
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
int main() {
    printf("This is execv.c called by execv()");
    printf("\n");
    return 0;
}
```

execv_main.c:

```c
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
int main() {
    char *args[]={"./execv",NULL};
    execv(args[0],args);
    printf("end of program.\n");
    return 0;
}
```

```
vishwa@Acer-Vishwa:/mnt/c/Users/Vishwa/Documents/Vishwa_PES/Sem4/OS/OS_HandsOn/week2$ ./execv_main
This is execv.c called by execv()
vishwa@Acer-Vishwa:/mnt/c/Users/Vishwa/Documents/Vishwa_PES/Sem4/OS/OS_HandsOn/week2$
```

Program 4:

cd, ps, du, w, kill

```
vishwa@Acer-Vishwa:/mnt/c/Users/Vishwa/Documents/Vishwa_PES/Sem4/OS/OS_HandsOn/week2$ cd .
vishwa@Acer-Vishwa:/mnt/c/Users/Vishwa/Documents/Vishwa_PES/Sem4/OS/OS_HandsOn/week2$ cd ..
vishwa@Acer-Vishwa:/mnt/c/Users/Vishwa/Documents/Vishwa_PES/Sem4/OS/OS_HandsOn$ cd week2/
vishwa@Acer-Vishwa:/mnt/c/Users/Vishwa/Documents/Vishwa_PES/Sem4/OS/OS_HandsOn/week2$ ps
  PID TTY          TIME CMD
    9 pts/0    00:00:00 bash
  427 pts/0    00:00:00 ps
vishwa@Acer-Vishwa:/mnt/c/Users/Vishwa/Documents/Vishwa_PES/Sem4/OS/OS_HandsOn/week2$ ps -a
  PID TTY          TIME CMD
  428 pts/0    00:00:00 ps
vishwa@Acer-Vishwa:/mnt/c/Users/Vishwa/Documents/Vishwa_PES/Sem4/OS/OS_HandsOn/week2$ du
0       .
vishwa@Acer-Vishwa:/mnt/c/Users/Vishwa/Documents/Vishwa_PES/Sem4/OS/OS_HandsOn/week2$ w
 13:55:21 up  6:12,  0 users,  load average: 0.00, 0.00, 0.08
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
```

```
vishwa@Acer-Vishwa:/mnt/c/Users/Vishwa/Documents/Vishwa_PES/Sem4/OS/OS_HandsOn/week2$ echo helloworld
helloworld
vishwa@Acer-Vishwa:/mnt/c/Users/Vishwa/Documents/Vishwa_PES/Sem4/OS/OS_HandsOn/week2$ ps
  PID TTY          TIME CMD
    9 pts/0    00:00:00 bash
  754 pts/0    00:00:00 ps
vishwa@Acer-Vishwa:/mnt/c/Users/Vishwa/Documents/Vishwa_PES/Sem4/OS/OS_HandsOn/week2$ ps -a
  PID TTY          TIME CMD
  741 pts/1    00:00:00 vim
  755 pts/0    00:00:00 ps
vishwa@Acer-Vishwa:/mnt/c/Users/Vishwa/Documents/Vishwa_PES/Sem4/OS/OS_HandsOn/week2$ kill -9 741
vishwa@Acer-Vishwa:/mnt/c/Users/Vishwa/Documents/Vishwa_PES/Sem4/OS/OS_HandsOn/week2$ ps -a
  PID TTY          TIME CMD
  756 pts/0    00:00:00 ps
vishwa@Acer-Vishwa:/mnt/c/Users/Vishwa/Documents/Vishwa_PES/Sem4/OS/OS_HandsOn/week2$ ps -n
  PID TTY      STAT   TIME COMMAND
    9 pts/0    Ss     0:00 -bash
  436 pts/1    Ss+    0:00 -bash
  757 pts/0    R+     0:00 ps -n
vishwa@Acer-Vishwa:/mnt/c/Users/Vishwa/Documents/Vishwa_PES/Sem4/OS/OS_HandsOn/week2$
```