 ***DEPARTMENT OF COMPUTER ENGINEERING***

Experiment No.

|  |  |
| --- | --- |
| Semester | S.E. Semester IV – Computer Engineering |
| Subject | Operating System |
| Subject Professor In-charge | Snehal Andhare |
| Assisting Teachers | Ms. Rasika Ransing |
| Laboratory | M310B – Computer Engineering Laboratory |

|  |  |  |
| --- | --- | --- |
| Student Name | Chinmay Tiwari | |
| Roll Number | 18102A0066 | |
| Grade and Subject Teacher’s Signature |  |  |

|  |  |  |
| --- | --- | --- |
| Experiment Number |  | |
| Experiment Title | Create a child process in Linux using the fork system call. From the child process obtain the process ID of both child and parent by using getpid and getppid system call. Explore wait and waitpid before termination of process. | |
| Resources / Apparatus Required | Hardware:  PC | Software:  Compilers |
| Objectives  (Skill Set / Knowledge Tested / Imparted) |  | |
| Theory: | **fork() function**  Fork system call is used for creating a new process, which is called child process, which runs concurrently with the process that makes the fork() call (parent process). After a new child process is created, both processes will execute the next instruction following the fork() system call. A child process uses the same pc(program counter), same CPU registers, same open files which use in the parent process.  **getpid()** returns the process ID (PID) of the calling process. (This is often used by routines that generate unique temporary filenames.)  **getppid()** returns the process ID of the parent of the calling process | |
| Code: | #include<stdio.h>  #include<sys/types.h>  #include<unistd.h>  int main()  {  int pid;  pid=fork();  pid\_t process\_id;  pid\_t p\_process\_id;  process\_id=getpid();  p\_process\_id=getppid();  printf("father id=%d child id=%d ",p\_process\_id,process\_id);  }  #include<stdio.h>  #include<sys/wait.h>  #include<unistd.h>  int main()  {  if (fork()== 0)  printf("HC: hello from child\n");  else  {  printf("HP: hello from parent\n");  wait(NULL);  printf("CT: child has terminated\n");  }  printf("Bye\n");  return 0;  }  #include<stdio.h>  #include<sys/wait.h>  #include<sys/types.h>  #include<unistd.h>  #include<stdlib.h>  const int NPROCESSES = 32;  pid\_t pids[128];  void main()  {  for(int i=0;i<NPROCESSES;i++)  {  pids[i] = fork();  if (!pids[i])  {  exit(0);  }  }  for (int i=0;i<NPROCESSES;i++)  {  printf("\tID = %d",pids[i]);  waitpid(pids[i],0,0);  }  } | |
| Output |  | |