 ***DEPARTMENT OF COMPUTER ENGINEERING***

Experiment No.

|  |  |
| --- | --- |
| Semester | S.E. Semester IV – Computer Engineering |
| Subject | Operating System |
| Subject Professor In-charge | SNA |
| 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 | Explore the following system calls: open, read, write, close, getpid, setpid, getuid, getgid, getegid, geteuid. | |
| Resources / Apparatus Required | Hardware: PC | Software:  Compiler |
| Objectives  (Skill Set / Knowledge Tested / Imparted) | System calls | |
| Theory: | In computing, a system call is the programmatic way in which a computer program requests a service from the kernel of the operating system it is executed on. This may include hardware-related services (for example, accessing a hard disk drive), creation and execution of new processes, and communication with integral kernel services such as process scheduling. System calls provide an essential interface between a process and the operating system.  In most systems, system calls can only be made from userspace processes, while in some systems, OS/360 and successors for example, privileged system code also issues system calls. | |
| Code | // open, read, write, close  #include<stdio.h>  #include<string.h>  #include<unistd.h>  #include<fcntl.h>    int main (void)  {  int fd[2];  char buf1[12] = "hello world";  char buf2[12];    fd[0] = open("foobar.txt", O\_RDWR);  fd[1] = open("foobar.txt", O\_RDWR);    write(fd[0], buf1, strlen(buf1));  write(1, buf2, read(fd[1], buf2, 12));    close(fd[0]);  close(fd[1]);    return 0;  }  //getpid, getuid, getgid, getegid, geteuid.  #include<stdio.h>  #include<sys/types.h>  #include<unistd.h>  int main()  {  int pid;  pid\_t process\_id;  process\_id=getpid();  printf("getpid() = %d\n",process\_id);  process\_id=getuid();  printf("getuid() = %d\n",process\_id);  process\_id=getgid();  printf("getgid() = %d\n",process\_id);  process\_id=getegid();  printf("getegid() = %d\n",process\_id);  process\_id=geteuid();  printf("geteuid() = %d\n",process\_id);  } | |
| Output |  | |