Experiment 09

Aim: Read( ) & Write( ) system calls in shell scripting.

1) Write System call

vi syscalls.c

```
#include <stdio.h>
#include <unistd.h>
int main() {
   int count;
   count = write(1, "hello\n", 6);
   printf("Total bytes written: %d\n",
count);
   return 0;
}

:wq

gcc syscalls.c -o syscalls

./syscalls
```

```
localhost:~/aatif# vi syscalls.c

#include <stdio.h>
#include <unistd.h>

int main() {
    int count;
    count = write(1, "hello\n", 6);
    printf("Total bytes written: %d\n", count);
    return 0;
}

~
~
:wq
localhost:~/aatif# gcc syscalls.c -o syscalls
localhost:~/aatif# ./syscalls
hello
Total bytes written: 6
```

2) Read and Write System Call

vi read.c

```
#include <stdio.h>
#include <unistd.h>
int main() {
   int nread;
   char buff[20];
   nread = read(0, buff, 10);
   write(1, buff, 10);

   return 0;
}

:wq

gcc read.c -o read

./read
```

```
localhost:~/aatif# vi read.c
#include <stdio.h>
#include <unistd.h>

int main() {
    int nread;
    char buff[20];

    // Read 10 bytes from standard input
    nread = read(0, buff, 10);

    // Write the read bytes to standard output
    write(1, buff, 10);

    return 0;
}

~
~
:wq
localhost:~/aatif# gcc read.c -o read
localhost:~/aatif# ./read
qwertyuiop
qwertyuioplocalhost:~/aatif#
localhost:~/aatif# ./read
qwertyuiopa
qwertyuioplocalhost:~/aatif# a
sh: a: not found
```

3) Fork System Call

vi fork.c

```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>

int main() {
   pid_t p;
   printf("Before fork\n");

   p = fork();

   if (p == 0) {
      printf("I am child having ID:
%d\n", getpid());
      printf("My parent ID is:
%d\n", getppid());
   }
   else {
      printf("My child ID is:
%d\n", p);
      printf("I am parent having
ID: %d\n", getpid());
   }

   printf("Common\n");

   return 0;
}
:wq
```

```
gcc fork.c -o fork

./fork
```

```
localhost:~/aatif# vi fork.c
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>

int main() {
    pid_t p;
    printf("Before fork\n");

    p = fork();

    if (p == 0) {
        printf("I am child having ID: %d\n", getpid());
        printf("My parent ID is: %d\n", getppid());
    }
    else {
        printf("My child ID is: %d\n", p);
        printf("I am parent having ID: %d\n", getpid());
    }

    printf("Common\n");

    return 0;
}
~
~
:wq
```

```
localhost:~/aatif# gcc fork.c -o fork
localhost:~/aatif# ./fork
Before fork
My child ID is: 93
I am parent having ID: 92
Common
I am child having ID: 93
My parent ID is: 92
Common
```