VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI

SWE3001 – Operating Systems Laboratory Manual

Lab - 02

Student Name	Sam Prince Franklin
Reg Number	20MIS1115
Subject Code	SWE3001
Slot	L37,38
Faculty	Dr.M.Braveen

Sam Prince Franklin K **20MIS1115**

SWE3001 – Operating Systems

Lab – 02 – System Calls

System Calls :

A system call is a method for a computer program to request a service from the kernel of the operating system on which it is running. A system call is a method of interacting with the operating system via programs. A system call is a request from computer software to an operating system's kernel.

Fork ( ) :

A fork system call is used for creating a new process, which is called the 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, and same open files use in the parent process.

1) Example 1 :

```c
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
void parentchild()
{
    int x=1;
    if(fork()==0)
    printf("Hello form Child has x= %d\n",++x);
    else
    printf("Hello from PARENT has x= %d\n",--x);
}

int main()
{
    parentchild();
    return 0;
}
```

Output :

```
samprince@samprince:~/Desktop/SWE3001/Lab2$ gcc fork.c -o fork
samprince@samprince:~/Desktop/SWE3001/Lab2$ ./fork
Hello from PARENT has x= 0
Hello form Child has x= 2
```

2) Example 2 :

```c
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
void parentchild()
{
    int x=1;
    if(fork()==0)
    printf("Hello form Child has x= %d\n",x);
    else
    printf("Hello from PARENT has x= %d\n",x);
}

int main()
{
    parentchild();
    return 0;
}
```

Output :

```
samprince@samprince:~/Desktop/SWE3001/Lab2$ gcc fork1.c -o fork1
samprince@samprince:~/Desktop/SWE3001/Lab2$ ./fork1
Hello from PARENT has x= 1
Hello form Child has x= 1
```

3) Example 3:

```c
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
int main()
{
    fork();
        fork();
            fork();
            printf("Welcome\n");
}
```

Output :

```
samprince@samprince:~/Desktop/SWE3001/Lab2$ gcc fork2.c -o fork2
samprince@samprince:~/Desktop/SWE3001/Lab2$ ./fork2
Welcome
Welcome
Welcome
Welcome
Welcome
Welcome
Welcome
Welcome
```

4) Example 4 :

```c
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
void forksleep()
{
    int x=1;
    if(fork()==0)
    printf("Hello form Child has x= %d\n",++x);
    else
    printf("Hello from PARENT has x= %d\n",--x);
}

int main()
{
    forksleep();
    sleep(2);
    forksleep();
    return 0;
}
```

Output :

```
samprince@samprince:~/Desktop/SWE3001/Lab2$ gcc fork3.c -o fork3
samprince@samprince:~/Desktop/SWE3001/Lab2$ ./fork3
Hello from PARENT has x= 0
Hello form Child has x= 2
Hello from PARENT has x= 0
Hello form Child has x= 2
Hello from PARENT has x= 0
Hello form Child has x= 2
```

5) Take input and find if the given number is even/odd using child process and if its palindrome using parent process.

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

void oddevenpal()
{
    int num,r,sum=0,temp,num1;
    scanf("%d",&num);
    temp=num;
    if(fork()==0)
    {    if(num % 2==0)
            printf("%d is even\n",num);
            else
            printf("%d is odd \n",num);
    }
    //palindrome
    else
    {
       while(num>0)
        {
            r=num%10;
            sum=(sum*10)+r;
            num=num/10;
        }
        if(temp==sum)
            printf("%d is palindrome number. \n",temp);
        else
            printf("%d is not palindrome number. \n",temp);
    }
}

int main()
{
    oddevenpal();
    return 0;
}
```

Output :

```
samprince@samprince:~/Desktop/SWE3001/Lab2$ gcc forkex.c -o forkex
samprince@samprince:~/Desktop/SWE3001/Lab2$ ./forkex
34
34 is not palindrome number.
34 is even
samprince@samprince:~/Desktop/SWE3001/Lab2$ ./forkex
22
22 is palindrome number.
22 is even
```

6) Implement the Krishnamurthy Number detection using fork concepts of system calls.

```c
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
void knum()
{
    int num, i, rem, temp, fact, sum = 0;

    printf("Enter a number: ");
    scanf("%d", &num);

    for (temp = num; temp > 0; temp = temp/10){
        fact = 1;
        rem = temp % 10;

        for (i = 1; i <= rem; i++){
            fact = i * fact;
        }
        sum = sum + fact;
    }
    if(fork()==0)

    if (num == sum){
        printf("%d is a krishnamurthy number.", num);
    }
    else{
        printf("%d is not a krishnamurthy number.", num);
    }

}


int main()
{
    knum();

    return 0;
}
```

Output :

```
samprince@samprince:~/Desktop/SWE3001/Lab2$ gcc knum.c -o knum
samprince@samprince:~/Desktop/SWE3001/Lab2$ ./knum
Enter a number: 23
23 is not a krishnamurthy number.
```