College of Engineering,Trivandrum

Experiment 2

Network Programming Lab

Author:
Alan Anto

Registration Number :
TVE16CS09

February 4, 2019

Contents

1 System Calls

1.1 AIM

Familiarizing and understanding the use and functioning of System Calls used for Operating system and network programming in Linux

1.2 Theory

System calls are used by the programs to interact with the Operating System. Application Programme Interfaces (APIs) are used by the user programmes to access these system calls.

Services Provided by System Calls :

1.Process creation and management

2.Main memory management

3.File Access, Directory and File system management

4.Device handling(I/O)

5.Protection

6.Networking, etc.

1.3 Process Control

1.3.1 Create Process

fork() is used to create a process . A child process, which is a copy of the parent process is created every time fork() is called.The fork() call returns the PID of the child process created if it is called successfully, else it returns a negative number.

fork.c

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

int main()
{
    fork();
    printf("Using fork() system call\n");
    return 0;
}
```

```
Alans-MacBook-Air:exp2 alan$ gcc fork.c
Alans-MacBook-Air:exp2 alan$ ./a.out
Using fork() system call
Using fork() system call
Alans-MacBook-Air:exp2 alan$
```

1.3.2 Exit Process

exit() is used to terminate a process. An exit code is also passed as an argument to the same.

```c
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    printf("START");

    exit(0);

    printf("End of program");
}
```

```
Alans-MacBook-Air:exp2 alan$ gcc exit.c
Alans-MacBook-Air:exp2 alan$ ./a.out
STARTAlans-MacBook-Air:exp2 alan$
```

1.4 Wait

It is used for blocking the calling process until one of its child processes exits or a signal is received. The system call returns the PID of the terminated child on success and -1 on error.

wait.c

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

int main()
{
    pid_t cpid;
    if (fork()== 0)
        exit(0);
    else
        cpid = wait(NULL);
    printf("Parent pid = %d\n", getpid());
    printf("Child pid = %d\n", cpid);

    return 0;

}
```

```
Alans-MacBook-Air:exp2 alan$ gcc wait.c
Alans-MacBook-Air:exp2 alan$ ./a.out
Parent pid = 75370
Child pid = 75371
Alans-MacBook-Air:exp2 alan$
```

1.5 File Management

1.5.1 Create File

The open() system call is used to create a new file or open an existing file to read data from or write data to. The system call returns the file descriptor of the new file or -1 if an error has occurred.

open.c

```c
#include<stdio.h>
#include<fcntl.h>
#include<errno.h>
extern int errno;
int main()
{

    int fd = open("sampletext.txt", O_RDONLY | O_CREAT);

    printf("fd = %d/n", fd);

    if (fd ==-1)
    {

        printf("Error Number % d \n", errno);


        perror("Program");
    }
    return 0;
}
```

```
~/Downloads/S6/Networking Lab/exp2 —
[Alans-MacBook-Air:exp2 alan$ gcc open.c
[Alans-MacBook-Air:exp2 alan$ ./a.out
fd = 3/nAlans-MacBook-Air:exp2 alan$
```

1.5.2 Write File

The write() system call is used to write data from a file opened using the open() call. The system call returns the number of bytes successfully written to the file and the file pointer is also moved accordingly.

write.c

```c
#include <unistd.h>

int main(void)
{
    if (write(1, "This will be output to standard out\n", 36) != 36) {
        write(2, "There was an error writing to standard out\n", 44);
        return -1;
    }

    return 0;
}
```

```
[Alans-MacBook-Air:exp2 alan$ gcc write.c
[Alans-MacBook-Air:exp2 alan$ ./a.out
This will be output to standard out
Alans-MacBook-Air:exp2 alan$
```

1.5.3 Close Handle

The close() system call is used to close a open file.The call returns zero on success and -1 on error. Once closed, all locks held on that file by that process are also

released.

1.6 Communication

1.6.1 Create Pipe

The pipe() system call is used to create an unnamed pipe. The created pipe is represented by two file descriptors, one to which data is written and one from which data is read. Thus, the pipe is unidirectional.The system call returns zero on success and -1 on error.

pipe.c

```c
#include <stdio.h>
#include <unistd.h>
#define MSGSIZE 16
char* msg1 = "hello, world #1";
char* msg2 = "hello, world #2";
char* msg3 = "hello, world #3";

int main()
{
    char inbuf[MSGSIZE];
    int p[2], i;

    if (pipe(p) < 0)
        exit(1);



    write(p[1], msg1, MSGSIZE);
    write(p[1], msg2, MSGSIZE);
    write(p[1], msg3, MSGSIZE);

    for (i = 0; i < 3; i++) {

        read(p[0], inbuf, MSGSIZE);
        printf("% s\n", inbuf);
    }
    return 0;
}
```

```
[Alans-MacBook-Air:exp2 alan$ gcc pipe.c
pipe.c:14:9: warning: implicitly declaring library function 'exit' with type 'vo
        __attribute__((noreturn))' [-Wimplicit-function-declaration]
        exit(1);
        ^
pipe.c:14:9: note: include the header <stdlib.h> or explicitly provide a declara
      'exit'
pipe.c:25:18: warning: flag ' ' results in undefined behavior with 's' conversio
      [-Wformat]
        printf("% s\n", inbuf);
                 ~^~
2 warnings generated.
[Alans-MacBook-Air:exp2 alan$ ./a.out
hello, world #1
hello, world #2
hello, world #3
Alans-MacBook-Air:exp2 alan$
```

1.6.2 Create File Mapping

The shmget() system call is used to create a shared memory segment. The memory segment has an associated key which is passed to the system call along with the size of the memory segment and some control flags. The system call returns a segment identifier on success and -1 on error.

shmget.cpp

```cpp
#include <iostream>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <stdio.h>
using namespace std;

int main()
{

    key_t key = ftok("shmfile",65);


    int shmid = shmget(key,1024,0666|IPC_CREAT);


    char *str = (char*) shmat(shmid,(void*)0,0);

    cout<<"Write Data : ";
    gets(str);

    printf("Data written in memory: %s\n",str);


    shmdt(str);

    return 0;
}
```

```
Alans-MacBook-Air:exp2 alan$ g++ shmget.cpp
Alans-MacBook-Air:exp2 alan$ ./a.out
warning: this program uses gets(), which is unsafe.
Write Data : hi
Data written in memory: hi
Alans-MacBook-Air:exp2 alan$
```

1.7 Information Maintenance

1.7.1 Current Process ID

The getpid() system call returns the pid of the process which called it.

1.7.2 SET TIMER

The alarm() system call is used to generate a signal after a specified amount of time has elapsed.The call returns the number of seconds remaining until any previously created alarm is due to be generated.

1.7.3 Sleep

The sleep system call causes the calling process to be suspended until the specified time in seconds has elapsed.

1.8 Result

The familiarization of system calls was completed successfully.