**ASSIGNMENT # 2**



**Fall 2020**

**CSE302 System Programming**

Submitted by: **Shah Raza**

Registration No. : **18PWCSE1658**

Class Section: **B**

“On my honor, as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work.”

Student Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Submitted to:

**Prof. Tariq Kamal**

Thursday, February 25th, 2021

**Department of Computer Systems Engineering**

**University of Engineering and Technology, Peshawar**

**Question #1:**

**Compare and contrast select and poll system calls. What is the maxfd field (first parameter to the select call) and what it has to do with the performance of select? Does the poll system call perform better than select? if yes, why or when? If no, why not?**

**Answer:**

**Comparison:**

The Select system call organizes information by the type of event and has separate descriptor masks for read, write and error conditions. In contrast, the poll system call organizes the information by file descriptor rather than by the type of condition. The possible events are stored in a struct pollfd.

**Maxfd and its effect on performance of select:**

All the files opened by a process is listed in the file descriptor table. In the FDT, each file is assigned a file descriptor number. To improve the performance of select system call, the value of maxfd should be the maximum file descriptor number that we want to monitor plus one. This will ensure that we don’t go beyond the desired fd number.

**Does the poll system call perform better than select?**

Yes, the poll system call performs better than select. In select system call, we need to provide maxfd, if we provide the wrong maxfd then it can lead to performance issues while in poll system call we don’t need maxfd.

**Question #2:**

**Write a C program that implements the nprintf () function. nprintf is same as printf with one difference: it does not buffer the strings passed to it for writing to console. For doing this you are not allowed to use the printf/putc function calls in your program.**

**Code:**

/\*

    Name: Shah Raza

    Assignment no: 2

    Question #2

\*/

#include <stdio.h>

#include <unistd.h>

#include <stdarg.h>

#include <string.h>

/\*

    Since we are not allowed to use printf, we need some other means to print to

    the console. And for that I am using write() system call. The problem is that

    write() system call can only be used to print strings. So if we want to print an

    integer then we need to first convert it to string.

\*/

char \*ConvertToString(int num, int base)

{

    static char Representation[]= "0123456789ABCDEF";

    static char buffer[50];

    char \*ptr;

    ptr = &buffer[49];

    \*ptr = '\0';

    do

    {

        \*--ptr = Representation[num%base];

        num /= base;

    }while(num != 0);

    return(ptr);

}

//nprintf will work the same as printf without buffering

void nprintf(char \*format,...)

{

    char \*chp;

    char \*string;

    int num;

    char ch;

    va\_list arguments;

    va\_start(arguments,format);

    chp = format;

    while(\*chp!='\0')

    {

        if(\*chp=='%')

        {

            chp++;

            switch(\*chp)

            {

                case 'd':

                    num = va\_arg(arguments,int);

                    if(num<0)

                    {

                        num=-num;

                        ch = '-';

                        write(STDOUT\_FILENO,&ch,1);

                        //fflush will clear stdout's buffer

                        fflush(stdout);

                    }

                    string = ConvertToString(num,10);

                    write(STDOUT\_FILENO,string,strlen(string));

                    fflush(stdout);

                    break;

                case 's':

                    string = va\_arg(arguments,char \*);

                    write(STDOUT\_FILENO,string,strlen(string));

                    fflush(stdout);

                    break;

                case 'c':

                    ch = va\_arg(arguments,int);

                    write(STDOUT\_FILENO,&ch,1);

                    fflush(stdout);

                    break;

            }

        }

        else

        {

        write(STDOUT\_FILENO,chp,1);

        fflush(stdout);

        }

        chp++;

    }

    va\_end(arguments);

}

int main()

{

    char \*s = "SHAH RAZA";

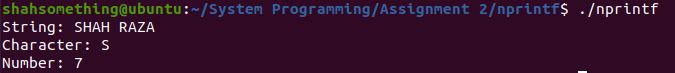
    char ch = 'S';

    int num = 7;

    nprintf("String: %s\nCharacter: %c\nNumber: %d\n",s,ch,num);

}

**Output:**



**Question #3:**

**Write a C program that implements the Linux diff command.**

**$ diff file1.txt file2.txt**

**Code:**

#include<stdio.h>

#include<string.h>

struct lines{

    char stack\_lines1[100];

    char stack\_lines2[100];

};

int main(int argc,char \*argv[])

{

    struct lines st[20];

    FILE \*file1,\*file2;

    char line1[100],line2[100];

    int line\_count1=0,flag2[10],flag1[10],line\_count2=0;

    int i=0,j=0,k=0,n=0,m=0,o=0;

    file1 = fopen(argv[1],"r");

    file2 = fopen(argv[2],"r");

    while(1)

    {

        line\_count1++;

        line\_count2++;

        if(fscanf(file1,"%[^\n]\n",line1)!=EOF && fscanf(file2,"%[^\n]\n",line2)!=EOF)

        {

            if(strcmp(line1,line2) == 0)

            continue;

            else{

                if(line1 != NULL){

                    strcpy(st[i].stack\_lines1 , line1);

                    flag1[m]=line\_count1;

                    m++;

                }

                if(line2 != NULL){

                    strcpy(st[i].stack\_lines2 , line2);

                    flag2[o]=line\_count2;

                    o++;

                }

            }

        }

        else if(fscanf(file1,"%[^\n]\n",line1)!=EOF){

            strcpy(st[i].stack\_lines1 , line1);

            flag1[m]=line\_count1;

            m++;

        }

        else if(fscanf(file2,"%[^\n]\n",line2)!=EOF){

            strcpy(st[i].stack\_lines2 , line2);

            flag2[o]=line\_count2;

            o++;

        }

        else

            break;

        i++;

        n++;

    }

    for(i=0;i<m;i++)

    {

        printf("%d,",flag1[i]);

    }

    printf("c");

    for(i=0;i<o;i++)

    {

        printf("%d,",flag2[i]);

    }

    printf("\n");

    for(i=0;i<n;i++)

    {

        printf("%s\n",st[i].stack\_lines1);

    }

    printf("---\n");

    for(i=0;i<n;i++)

    {

        printf("%s\n",st[i].stack\_lines2);

    }

    fclose(file1);

    fclose(file2);

    return 0;

}

**File1.txt:**

Team Pakistan:

Sharjeel Khan

Rohit Sharma

Babar Azam

Haider Ali

Muhammad Hafeez

Rishab Pant

Shadab Khan

Hardik Pandya

Faheem Ashraf

Jasprit Bumrah

Shaheen Shah Afridi

**File2.txt:**

Team Pakistan:

Sharjeel Khan

Muhammad Rizwan

Babar Azam

Haider Ali

Muhammad Hafeez

Asif Ali

Shadab Khan

Imad Wasim

Faheem Ashraf

Hassan Ali

Shaheen Shah Afridi

**Output:**

