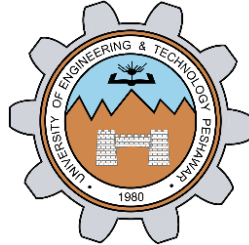


LAYOUT OF PROGRAM, LIBRARY FUNCTION

CALLS AND ERROR

HANDLING

LAB # 02



Fall 2024

CSE-302L

Systems Programming Lab

Submitted by: **AIMAL KHAN**

Registration No.: **21PWCSE1996**

Class Section: **A**

“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:

Engr. Abdullah Hamid

Department of Computer Systems Engineering
University of Engineering and Technology, Peshawar

CSE 302L: SYSTEMS PROGRAMMING LAB

LAB ASSESSMENT RUBRICS

Criteria & Point Assigned	Outstanding 2	Acceptable 1.5	Considerable 1	Below Expectations 0.5	Score
Attendance and Attentiveness in Lab PLO08	Attended in proper Time and attentive in Lab	Attended in proper Time but not attentive in Lab	Attended late but attentive in Lab	Attended late not attentive in Lab	
Capability of writing Program/Algorithm/Drawing Flow Chart PLO1, PLO2, PLO3, PLO5	Right attempt/ no errors and well formatted	Right attempt/ no errors but not well formatted	Right attempt/ minor errors and not well formatted	Wrong attempt	
Result or Output/ Completion of target in Lab PLO9	100% target has been completed and well formatted.	75% target has been completed and well formatted.	50% target has been completed but not well formatted.	None of the outputs are correct.	
Overall, Knowledge PLO10,	Demonstrates excellent knowledge of lab	Demonstrates good knowledge of lab	Has partial idea about the Lab and procedure followed	Has poor idea about the Lab and procedure followed	
Attention to Lab Report PLO4,	Submission of Lab Report in Proper Time i.e., in next day of lab, with proper documentation.	Submission of Lab Report in proper time but not with proper documentation.	Late Submission with proper documentation.	Late Submission very poor documentation	

Instructor:

Name: _____	Signature: _____
-------------	------------------

Layout of Program, Library Function Calls and Error Handling

Objectives:

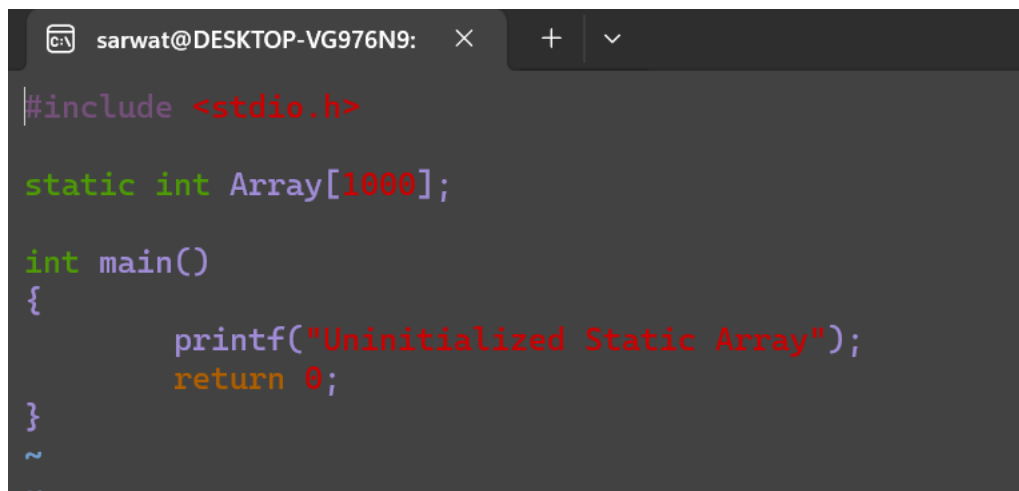
In this lab we will learn about Layout of Program, Library Function Calls and error handling.

Tasks:

Task 1: . Write two C programs for the analysis of the difference in executable file sizes while using:

1. Uninitialized Static Variables
2. Initialized Static Variables

1. Uninitialized Static Variables

A screenshot of a code editor window titled 'sarwat@DESKTOP-VG976N9:'. The code is a C program that includes <stdio.h>, declares a static integer array of size 1000, and prints 'Uninitialized Static Array' in the main function.

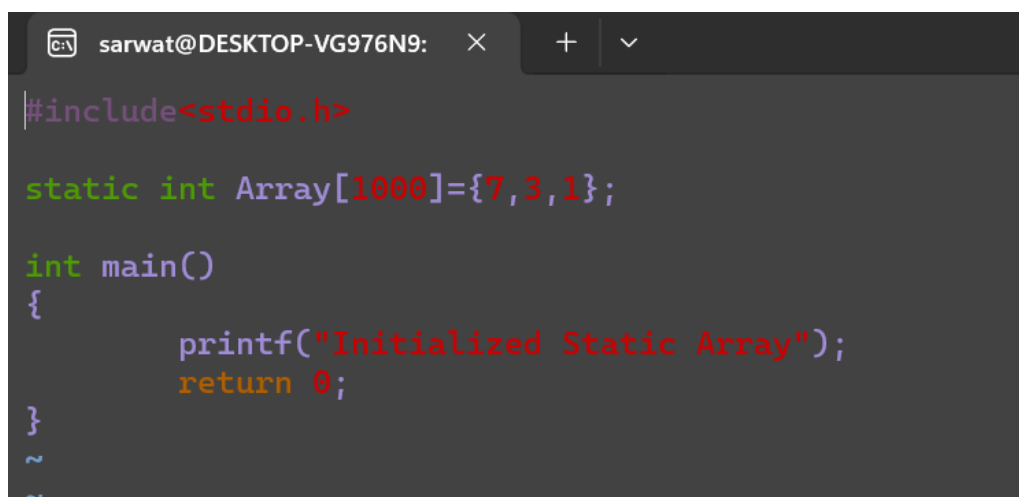
```
#include <stdio.h>

static int Array[1000];

int main()
{
    printf("Uninitialized Static Array");
    return 0;
}

~
~
```

2. Initialized Static Variables

A screenshot of a code editor window titled 'sarwat@DESKTOP-VG976N9:'. The code is a C program that includes <stdio.h>, declares a static integer array of size 1000 initialized with {7, 3, 1}, and prints 'Initialized Static Array' in the main function.

```
#include <stdio.h>

static int Array[1000]={7,3,1};

int main()
{
    printf("Initialized Static Array");
    return 0;
}

~
~
```

Output:

```
sarwat@DESKTOP-VG976N9: ~/lab2$ gcc part1.c -o part1.o
sarwat@DESKTOP-VG976N9: ~/lab2$ gcc part2.c -o part2.o
sarwat@DESKTOP-VG976N9: ~/lab2$ ls -l
total 108
-rw-r--r-- 1 sarwat sarwat 119 Jan 30 01:05 part1.c
-rwxr-xr-x 1 sarwat sarwat 15992 Jan 30 01:06 part1.o
-rw-r--r-- 1 sarwat sarwat 124 Jan 30 01:06 part2.c
-rwxr-xr-x 1 sarwat sarwat 20008 Jan 30 01:06 part2.o
-rw-r--r-- 1 sarwat sarwat 946 Oct 18 22:49 t2.c
-rwxr-xr-x 1 sarwat sarwat 16184 Oct 12 10:40 t2.o
drwxr-xr-x 4 sarwat sarwat 4096 Jan 30 01:04 task2
-rw-r--r-- 1 sarwat sarwat 346 Jan 30 00:52 task2.c
-rw-r--r-- 1 sarwat sarwat 0 Oct 12 09:46 task2.c.swp
-rwxr-xr-x 1 sarwat sarwat 16184 Oct 12 09:44 task2.o
-rw-r--r-- 1 sarwat sarwat 291 Oct 12 10:08 task3.c
-rwxr-xr-x 1 sarwat sarwat 16144 Oct 12 10:41 task3.o
sarwat@DESKTOP-VG976N9: ~/lab2$ |
```

Task 2: Analyze the return values and error numbers/error strings of wait system call on:

1. Success
2. Failure Using:
 1. Strerror
 2. Perror

1. Success

Strerror:

```

#include <stdio.h>
#include <errno.h>
#include <sys/wait.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>

int main()
{
    int fret= fork();
    if(fret==-1)
    {
        printf("Return value: %d\n",fret);
        printf("Error Occured: %s\n",strerror(errno));
        printf("Error No: %d\n",errno);
    }
    else if(fret==0)
    {
        exit(0);
    }
    else if(fret >0)
    {
        int ret = wait(NULL);
        if(ret==-1)
        {
            printf("Return value: %d\n",ret);
            printf("Error Occured: %s\n",strerror(errno));
            printf("Error No: %d\n",errno);
        }
        else
        {
            printf("Return value: %d\n",ret);
            printf("Operation Successful: %s\n",strerror(errno));
            printf("Error No: %d\n",errno);
        }
    }
}

```

```

sarwat@DESKTOP-VG976N9:~/lab2/task2/success$ ./strerror.o
Return value: 12109
Operation Successful: Success
Error No: 0
sarwat@DESKTOP-VG976N9:~/lab2/task2/success$ |

```

Perror:

```

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

int main()
{
    int fret= fork();
    if(fret==-1)
    {
        printf("Return value: %d\n",fret);
        perror("Error Occured");
        printf("Error No: %d\n",errno);
    }
    else if(fret==0)
    {
        exit(0);
    }
    else if(fret >0)
    {
        int ret = wait(NULL);
        if(ret==-1)
        {
            printf("Return value: %d\n",ret);
            perror("Error Occured");
            printf("Error No: %d\n",errno);
        }
        else
        {
            printf("Return value: %d\n",ret);
            perror("Operation Successful");
            printf("Error No: %d\n",errno);
        }
    }
}

```

```

sarwat@DESKTOP-VG976N9:~/lab2/task2/success$ ./perror.o
Return value: 19516
Operation Successful: Success
Error No: 0
sarwat@DESKTOP-VG976N9:~/lab2/task2/success$ |

```

Failure:

1: strerror

```
sarwat@DESKTOP-VG976N9: × + ▾  
#include <stdio.h>  
#include <errno.h>  
#include <string.h>  
#include <sys/wait.h>  
  
int main()  
{  
    int ret = wait(NULL);  
    printf("Return Value: %d\n",ret);  
    if(ret==-1)  
    {  
        printf("Error Occured: %s\n", strerror(errno));  
        printf("Error No: %d\n", errno);  
    }  
}
```

```
Return Value: -1  
Error Occured: No child processes  
Error No: 10  
sarwat@DESKTOP-VG976N9:~/lab2/task2/failure$ |
```

perror:

```
sarwat@DESKTOP-VG976N9: × + ▾  
#include <stdio.h>  
#include <sys/wait.h>  
#include <errno.h>  
  
int main()  
{  
    int ret = wait(NULL);  
    printf("Return Value: %d\n",ret);  
    if(ret==-1)  
    {  
        perror("Error Occured");  
        printf("Error No: %d \n",errno);  
    }  
}
```

Output:

```
sarwat@DESKTOP-VG976N9:~/lab2/task2/failure$ ./perror.o
Return Value: -1
Error Occured: No child processes
Error No: 10
sarwat@DESKTOP-VG976N9:~/lab2/task2/failure$ |
```

Task3:

Analyze the return values and error numbers/error strings of close system call on:

3. Success
4. Failure Using:
 3. Strerror
 4. Perror
- 1:success

Strerror:

```

sarwat@DESKTOP-VG976N9: × + v
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <errno.h>

int main()
{
    int ret = close(2);
    printf("Return value of close: %d\n",ret);
    if(ret==-1)
    {
        printf("Error Occured: %s\n",strerror(errno));
        printf("Error No: %d\n",errno);
    }
    else
    {
        printf("File closed Successfully: %s\n",strerror(errno));
        printf("Error No: %d\n",errno);
    }
}
```

Output:


```

sarwat@DESKTOP-VG976N9:~/lab2/task3$ cd success
sarwat@DESKTOP-VG976N9:~/lab2/task3/success$ gcc strerror.c -o strerror.o
sarwat@DESKTOP-VG976N9:~/lab2/task3/success$ ./strerror.o
Return value of close: 0
File closed Successfully: Success
Error No: 0
sarwat@DESKTOP-VG976N9:~/lab2/task3/success$ |

```

Perror:

```

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

int main()
{
    int ret = close(0);
    printf("Return value of close: %d\n",ret);
    if(ret==-1)
    {
        perror("Error Occured");
        printf("Error No: %d\n",errno);
    }
    else
    {
        perror("File Closed Successfully");
        printf("Error No: %d\n",errno);
    }
}

```

Output:

```

sarwat@DESKTOP-VG976N9:~/lab2/task3/success$ ./perror.o
Return value of close: 0
File Closed Successfully: Success
Error No: 0
sarwat@DESKTOP-VG976N9:~/lab2/task3/success$ |

```

2 Failure

Strerror:

```
sarwat@DESKTOP-VG976N9: X + v
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <errno.h>

int main()
{
    int ret = close(6);
    printf("Return value of close: %d\n",ret);
    if(ret==-1)
    {
        printf("Error Occured: %s\n",strerror(errno));
        printf("Error No: %d\n",errno);
    }
    else
    {
        printf("File closed Successfully.\n");
    }
}
```

Output:

```
sarwat@DESKTOP-VG976N9: X + v
sarwat@DESKTOP-VG976N9:~/lab2/task3/failure$ gcc strerror.c -o strerror.o
sarwat@DESKTOP-VG976N9:~/lab2/task3/failure$ ./strerror.o
Return value of close: -1
Error Occured: Bad file descriptor
Error No: 9
sarwat@DESKTOP-VG976N9:~/lab2/task3/failure$ |
```

Perror:

```
sarwat@DESKTOP-VG976N9: × + ∨  
#include <stdio.h>  
#include <unistd.h>  
#include <errno.h>  
  
int main()  
{  
    int ret = close(7);  
    printf("Return value of close: %d\n",ret);  
    if(ret==-1)  
    {  
        perror("Error Occured");  
        printf("Error No: %d\n",errno);  
    }  
    else  
    {  
        printf("File closed Successfully.\n");  
    }  
}
```

Output:

```
sarwat@DESKTOP-VG976N9:~/lab2/task3/failure$ ./perror.o  
Return value of close: -1  
Error Occured: Bad file descriptor  
Error No: 9  
sarwat@DESKTOP-VG976N9:~/lab2/task3/failure$ |
```

Conclusion:

In conclusion, I have learned about different library function calls.

The End.