National University of Computer and Emerging Sciences



Laboratory Manual

for

Operating Systems Lab

| Course Instructor | Dr Zeeshan Ali Khan |
| --- | --- |
| Lab Instructor(s) | Usman Anwer  Aiss Shahid |
| Section | BCS-F |
| Semester | Spring-2023 |

Department of Computer Science

FAST-NU, Lahore, Pakistan

**exec()**

The exec family of functions replaces the current running process with a new process. It can be used to run a C program by using another C program. It comes under the header file unistd.h.

There are many members in the exec family we will be seeing “excelp” for now.

The **execlp() function** replaces the current process image with a new process image specified by file. The new image is constructed from a regular, executable file called the new process image file. No return is made because the calling process image is replaced by the new process image.

**Syntax**

| #include <unistd.h>  int execlp( const char \* file,  const char \* arg0,  const char \* arg1,  …  const char \* argn,  NULL ); |
| --- |

**Arguments**

| **file**  Used to construct a pathname that identifies the new process image file. If the file argument contains a slash character, the file argument is used as the pathname for the file.  **arg0, …, argn**  Pointers to NULL-terminated character strings. These strings constitute the argument list available to the new process image. You must terminate the list with a NULL pointer. **The arg0 argument must point to a filename that's associated with the process being started and cannot be NULL.** |
| --- |

**Example**

Program to create image for

| #include <stdio.h>  #include <sys/types.h>  #include <unistd.h>  int main(int argc, char \* arg[])  {  printf(arg[1]);  return 0;  } |
| --- |

**Execlp program to run the Process**

| #include <stdio.h>  #include <sys/types.h>  #include <unistd.h>  int main(int argc, char \* arg[])  {  int x=execlp("./a.out","a.out","Hello",NULL); return 0;  } |
| --- |

**You can also use execlp to run unix system commands such as ls , cp , mkdir etc. Example is shown below**

**Example: creating a new directory named Lab using execlp**

| #include<stdlib.h>  #include<sys/wait.h>  #include<sys/types.h>  #include<unistd.h>  int main()  {  // creating a new directory named Lab using execlp  int x=execlp("mkdir","mkdir","LabTestFolder",NULL);    return 0;  } |
| --- |

**Question 1:** Write a C or C++ program that accepts a list of integers and a filename as command line arguments and the child finds the greatest, smallest and the average of the integers while parent creates a new file and stores the results computed by the child into that file. But there is a condition, if the sum of all the integers is less then 50, the process is terminated and parent print ‘NULL recieved’.

**Question 2:** Write a program that uses two processes. One is called a parent and the other is called a child. The child calls another C file named hello.c and executes the code written with in it.

The content of hello.c is,

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

int main(int argc, char \*argv[])

{

printf("We are in Hello.c\n");

return 0;

}

While child is processing, Parent wait for child to finish with code and prints ‘Program completed’ message.

Note: Use Fork and Exec system calls

Use the command line arguments to take the ‘hello.c’ filename as an input.

**Question 3: execlp( ) System Call Estimated Time: 30 mins**

**Using execlp () implement the following:**

1. **Create a directory named lab3-questtion3 (mkdir)**
2. **Create 2 new files in the directory you have created (touch)**
3. **List the files in the directory (ls)**
4. **Remove the directory**