CSC 33200 (L) - Operating Systems – Fall 2021

Lab 3: Process Management System Calls
Date: 10/08/2021

This handout describes the **exec** family of functions, for executing a comand. You can use these functionsto make a child process execute a new program after it has been forked. The functions in this family differ in how you specify the arguments, but otherwise they all do the same thing. They are declared in theheader file **'unistd.h'.**

execv (char *filename, char *const argv[])

The execvfunction executes the file named by filename as a new process image.

The argv argument is an array of null-terminated strings that is used to provide a value for the argv argument to the main function of the program to be executed. The last element of this array *must* be a null pointer.

execvp (char *filename, char *const argv[])

The execvp function is similar to execv, except that it searches the directories listed in the PATH environment variable to find the full file name of a file from filename if filename does not contain a slash.

This function is useful for executing system utility programs, because it looks for them in the places that the user has chosen. **Shells use** execvp **to run the commands** that users type.

execl (char *filename, const char *arg0, ...)

This is similar to execv, but the argv strings are specified individually instead of as an array. A null pointer must be passed as the last such argument.

execlp (char *filename, const char *arg0, ...)

This function is like execl, except that it performs the same file name searching as the execvp function.

Example 1: Using execv(...) command

Note: This version will not search the path, so the full name of the executable file must be given. Parameters to main() are passed in a single array of character pointers.

```
#include <unistd.h>

#include <stdio.h>


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


    execv("/bin/echo", &argv[0]);

    printf("EXECV Failed\n");


    return 0;

}
```

```
./example1.out "Hello World"
```

Example 2: Using execvp(...) command

Note: This version searches the path, so the full name of the executable need not be given. Parameters to main() are passed in a single array of character pointers. *This is the form used inside a shell!*

```
#include <unistd.h>

#include <stdio.h>


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


    execvp("echo", &argv[0]);

    printf("EXECVP Failed\n");


    return 0;

}
```

```
./example2.out “Hello World”
```

TASK 3. Marks: 20

Part 1 Write a program where a child is created to execute command that tells you the date and time inUnix.

Use execl(...).

Note, you need to specify the full path of the file name that gives you date and time information.Announce the successful forking of child process by displaying its PID.

Part 2 Write a program where a child is created to execute a command that shows all files (including hid-den files) in a directory with information such as permissions, owner, size, and when last modified.

Use execvp(...).

For the command to list directory contents with various options, refer the handout on Unix filesystem sent to you in the first class.

Announce the successful forking of child process by displaying its PID.

Part 3

[Step 1] Prcs_P1.c: Create two files namely, destination1.txt and destination2.txt with read, write,and execute permissions.

[Step 2] Prcs_P2.c: Copy the contents of source.txt[1] into destination1.txt and destination2.txt asper the following procedure.

1. Read the next 100 characters from source.txt, and among characters read, replace each char-acter '1' with character 'L' and all characters are then written in destination1.txt
2. Then the next 50 characters are read from source.txt, and among characters read, replace each character '3' with character 'E' and all characters are then written in destination2.txt.
3. The previous steps are repeated until the end of file source.txt. The last read may not have100 or 50 characters.

Once you're done with successful creation of executables for the above two steps do the following.

Write a C program and call it Parent_Prcs.c. Execute the files as per the following procedure using
execv system call. Use sleep system calls to introduce delays.

[Step 3] Using fork create a child process, say Child 1 that executes Prcs_P1. This will create two destinationfiles according to Step 1.

[Step 4] After Child 1 finishes its execution, use fork to create another child process, say Child 2 and execute
Prcs_P2 that accomplishes the procedure described in Step 2.

Submission Instructions

- All the programs MUST be clearly indented and internally documented
- Use the given source.txt
- Make sure your programs compile and run without any errors
- Save all your programs with meaningful names and zip into a single folder as: task3_[your last name here].zip (e.g., task3_Xyz.zip)
- Email your code with the subject line, "Task3-CSC33200(L)–Class# 12345-*lastname*" (e.g., Task3 - CSC33200(L)-Class #63858-Xyz)
- Email: sdebnath@ccny.cuny.edu

******

Office Hours: Thursday 10:00 – 11:00 pm

Join Zoom Meeting
https://ccny.zoom.us/j/85722468391?pwd=RG9sREJ2WUVBMzhsOWp6eWZHTmE3QT09

Meeting ID: 857 2246 8391
Passcode: 561041
One tap mobile
+16465588656,,85722468391# US (New York)
+13126266799,,85722468391# US (Chicago)
Dial by your location
+1 646 558 8656 US (New York)
+1 312 626 6799 US (Chicago)
+1 301 715 8592 US (Washington DC)
+1 669 900 6833 US (San Jose)
+1 253 215 8782 US (Tacoma)
+1 346 248 7799 US (Houston)
Meeting ID: 857 2246 8391

Find your local number: https://ccny.zoom.us/u/kek94YYHZu

IMPORTANT NOTE: Please connect with zoom with a valid CCNY or CITYMAIL email id. Please sign up with the CITYMAIL or CCNY email address at :
https://www.ccny.cuny.edu/it/zoom