Problem - 3

Name : SURLA VISHNU TEJA Roll: CS21B2037

exec() system call

exec(): This system call is used to replace the current process with a new process without actually creating another process and execute the new process.

The variants of exec are :-

(i) execl(): In this, the system call takes path of the executable file as first argument, and a variable number of argument strings, each passed individually, ending with 'NULL'

→ This is suitable when no'of arguments is known beforehand and arguments are known

→ The New process inherits current process environment

(ii) execv(): This, unlike execl, receives an array of strings as argument which contains all the the arguments. The last element of array must be 'NULL'

→ This can be used when no'of arguments may change dynamically

(iii) execle(): This extends the functionality of execl by allowing to pass environment variables along with arguments. The last 2 arguments are env variables and 'NULL'

(iv) execve() : An extension to the execv() system call that allows passage of environment variables

(v) execlp(): This allows all the programs in PATH environment variable specified directly without mentioning the entire path of program. This is an extension of execl().

(vi) execvp(): This is an extension of execv() that searches for a program in PATH environment variable without entire path

(vii) execvpe(): This is an extension of execvp() that allows search in PATH variable and sending in environment variables.

Name: SURLA VISHNU TEJA Roll: CS21B2037

wait() system call

wait() : This system call allows parent process to wait for the termination of it's child process and retrieve their exit status

The variants of wait are :-

(i) wait(): This suspends the execution until one of it's child process terminates and then returns the exit status, if input is not given as 'NULL'

(ii) waitpid(): This variant allows a parent process to wait for a specific child process to terminate by taking pid of child, a variable to store status of child, and options to add additional conditions as input

(iii) wait3() (or) wait4() : Old and depricated versions of waitpid() system call

```c
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>

int main() {
    execl("/bin/echo", "/bin/echo","Hello World!", NULL);
    return(0);
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL GITLENS SQL CONSOLE

```
(base) vishnu@pop-os:~/Desktop/Sem-5/Operating-System/Labs/Lab-5$ cd "/
3a.c -o Problem-3a && "/home/vishnu/Desktop/Sem-5/Operating-System/Labs
Hello World!
```

```c
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>

int main() {
    char *file = "/bin/echo";
    char *const args[] = {"/bin/echo","Hello World!", NULL};
    execv(file, args);
    return(0);
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL GITLENS SQL CONSOLE

```
(base) vishnu@pop-os:~/Desktop/Sem-5/Operating-System/Labs/Lab-5$ cd "/
3a.c -o Problem-3a && "/home/vishnu/Desktop/Sem-5/Operating-System/Labs
Hello World!
```

```c
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>

int main() {
    char *file = "/bin/bash";
    char *arg1 = "-c";
    char *arg2 = "echo $ENV1 $ENV2!";
    char *const env[] = {"ENV1=Hello","ENV2=Vishnu", NULL};
    execle(file, file, arg1, arg2, NULL, env);
    return(0);
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL GITLENS SQL CONSOLE

```
(base) vishnu@pop-os:~/Desktop/Sem-5/Operating-System/Labs/Lab-5$ cd "/h
3a.c -o Problem-3a && "/home/vishnu/Desktop/Sem-5/Operating-System/Labs/
Hello Vishnu!
```

```c
#include<unistd.h>

int main(void) {
    char *file = "/usr/bin/bash";
    char *const args[] = {"/usr/bin/bash", "-c", "echo Hello $ENV!", NULL};
    char *const env[] = {"ENV=World", NULL};

    execve(file, args, env);

    return 0;
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL GITLENS SQL CONSOLE

```
cd "/home/vishnu/Desktop/Sem-5/Operating-System/Labs/Lab-5/" && gcc Problem-3a.c -o
/Labs/Lab-5/"Problem-3a
(base) vishnu@pop-os:~/Desktop/Sem-5/Operating-System/Labs/Lab-5$ cd "/home/vishnu/D
3a.c -o Problem-3a && "/home/vishnu/Desktop/Sem-5/Operating-System/Labs/Lab-5/"Probl
Hello World!
```

```c
#include<unistd.h>

int main(void) {
    char *file = "echo";
    char *arg1 = "Hello world!";

    execlp(file, file, arg1, NULL);
    return 0;
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL GITLENS SQL CONSOLE

```
(base) vishnu@pop-os:~/Desktop/Sem-5/Operating-System/Labs/Lab-5$ cd "/home/vishn
3a.c -o Problem-3a && "/home/vishnu/Desktop/Sem-5/Operating-System/Labs/Lab-5/"P
Hello world!
```

```c
#include<unistd.h>

int main(void) {
    char *file = "echo";
    char *const args[] = {"/usr/bin/echo", "Hello world!", NULL};

    execvp(file, args);

    return 0;
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL GITLENS SQL CONSOLE

```
(base) vishnu@pop-os:~/Desktop/Sem-5/Operating-System/Labs/Lab-5$ cd "/home/vishn
3a.c -o Problem-3a && "/home/vishnu/Desktop/Sem-5/Operating-System/Labs/Lab-5/"P
Hello world!
```

```c
#include <unistd.h>

int main() {
    char *args[] = {"ls", "-l", NULL};
    char *env[] = {"MY_ENV=example", NULL};
    execvpe("ls", args, env);
    return 0;
}

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL GITLENS SQL CONSOLE

```
(base) vishnu@pop-os:~/Desktop/Sem-5/Operating-System/Labs/Lab-5$ cd "/home/vish
3a.c -o Problem-3a && "/home/vishnu/Desktop/Sem-5/Operating-System/Labs/Lab-5/"P
Problem-3a.c: In function ‘main’:
Problem-3a.c:6:5: warning: implicit declaration of function ‘execvpe’; did you m
    6 |     execvpe("ls", args, env);
      |     ^~~~~~~
      |     execvp
total 104
-rwxrwxr-x 1 vishnu vishnu 16336 Aug 25 14:41 Problem-1
-rw-rw-r-- 1 vishnu vishnu  2354 Aug 25 15:20 Problem-1.c
-rwxrwxr-x 1 vishnu vishnu 16344 Aug 25 15:15 Problem-2
-rw-rw-r-- 1 vishnu vishnu  2501 Aug 25 15:14 Problem-2.c
-rwxrwxr-x 1 vishnu vishnu 16016 Aug 25 17:20 Problem-3a
-rw-rw-r-- 1 vishnu vishnu   163 Aug 25 17:20 Problem-3a.c
-rwxrwxr-x 1 vishnu vishnu 16176 Aug 25 17:08 Problem-4
-rw-rw-r-- 1 vishnu vishnu  1673 Aug 25 17:08 Problem-4.c
-rwxrwxr-x 1 vishnu vishnu 15968 Aug 25 17:01 Problem-4b
-rw-rw-r-- 1 vishnu vishnu   103 Aug 25 17:01 Problem-4b.c
-rw-rw-r-- 1 vishnu vishnu    19 Aug 25 17:20 tempCodeRunnerFile.c
```

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

int main() {
    pid_t child_pid;

    if ((child_pid = fork()) == 0) {
        // Child process
        printf("Child process running...\n");
        sleep(2);
        printf("Child process completed.\n");
        exit(0);
    } else if (child_pid > 0) {
        // Parent process
        printf("Parent process waiting for child...\n");
        int status;
        wait(&status); // Parent waits for child to complete
        printf("Parent process resumed.\n");
    } else {
        perror("Fork failed");
        return 1;
    }

    return 0;
}
```

```
cd "/home/vishnu/Desktop/Sem-5/Operating-System/Labs/Lab-5/" && gcc Prob
abs/Lab-5/"Problem-3
(base) vishnu@pop-os:~/Desktop/Sem-5/Operating-System/Labs/Lab-5$ cd "/h
3.c -o Problem-3 && "/home/vishnu/Desktop/Sem-5/Operating-System/Labs/La
Parent process waiting for child...
Child process running...
Child process completed.
Parent process resumed.
```

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

int main() {
    pid_t child_pid;

    if ((child_pid = fork()) == 0) {
        // Child process
        printf("Child process running...\n");
        sleep(2);
        printf("Child process completed.\n");
        exit(0);
    } else if (child_pid > 0) {
        // Parent process
        printf("Parent process waiting for child...\n");
        int status;
        waitpid(child_pid, &status, 0); // Parent waits for a specific child
        printf("Parent process resumed.\n");
    } else {
        perror("Fork failed");
        return 1;
    }

    return 0;
}
```

```
cd "/home/vishnu/Desktop/Sem-5/Operating-System/Labs/Lab-5/" && gcc Prob
abs/Lab-5/"Problem-3
(base) vishnu@pop-os:~/Desktop/Sem-5/Operating-System/Labs/Lab-5$ cd "/h
3.c -o Problem-3 && "/home/vishnu/Desktop/Sem-5/Operating-System/Labs/La
Parent process waiting for child...
Child process running...
Child process completed.
Parent process resumed.
```