PES UNIVERSITY

(Established under Karnataka Act No.16 of 2013)

100-ft Ring Road, BSK III Stage, Bangalore – 560 085

Department of Computer Science

Session: Jan-May 2021

UE19CS254: Operating Systems

Assignment 1

1. Write a program to create a child process which calls one of the exec functions to create a file named "abc.txt"[Hint: shell command “touch filename” creates a new file]. Parent using one of the exec calls must execute the command "ls". Make the parent wait for the child to terminate and then execute "ls". Do not use "system" function

2. Write a program where in the child process initiates a new program which finds the sum of n numbers. The numbers to add are given as arguments in the **exec function**. Use appropriate exec function. Parent process should wait for the termination of child process.

3. Create a simple pipe between parent and child using pipe() system call. The child process writes information to the write end of the pipe and the parent process reads information from the read end of the pipe and display it to standard output device.

Questions

1. The process which terminates before the parent process exits and its termination status in not known to the parent process becomes_____.

a. **Zombie** b. Orphan c. Child d. Stopped Process

2. If fork() system call returns -1, then it means_____.

a. **No new child process is created**

b. The child process is an orphan

c. The child process is in Zombie

d. The child process is created.

3. Which system call returns the process identifier of a terminated child?

a) get b) exit c) fork d) **wait**

4. Which of the following is a primitive type of IPC

a. **pipes** b named pipes c. shared memory d. message queues

5.
```
int main()
{
if( execl("/bin/ls","ls",NULL) == -1){
printf("execl");
exit(1);
```

}
printf(“OS\n");
return 0;
}
Output of the program is_____
a. **Lists all the files in the current working directory** b. OS c. execl d. Nothing gets printed

Additional Material

fork() system call
fork (): Fork creates a new process (child process). It creates an exact duplicate of the original process, including all the file descriptors, registers etc. The fork is called once, but returns twice. After the fork, the original process and the copy (the parent and the child) go their separate ways. The fork call returns a value, which is zero in the child and equal to the child’s process identifier (PID) in the parent. Now consider how fork is used by the shell. When a command is typed, the shell forks off a new process. This child process must execute the user command.

Exec function: The exec() call replaces/overwrites a current process image with a new one (i.e. loads a new program within the current process). The file descriptor table remains the same as the original process. Argument passed via exec() appear in the argv[] of the main function. Upon success, exec() never returns to the caller. It replaces the current process image, so it cannot return anything to the program that made the call. If it does return, it means the call failed.
Ex: execl(“/bin/ls”,”ls”,NULL): overwrites the memory code image with binary from /bin/ls and execute.

exec variations
By exec() we usually refer to a family of calls:
int execl(char *path, char *arg, …); υ int execv(char *path, char *argv[]);
int execle(char *path, char *arg, …, char *envp[]);
int execve(char *path, char *argv[], char *envp[]);
int execlp(char *file, char *arg, …);
int execvp(char *file, char *argv[]);

Where: l=argument list, v= argument vector, e=environmental vector, p= search path

Wait(): Forces the parent to suspend execution, i.e. wait for its children or a specific child to die(terminate). When the child process terminates, it returns an exit status to the OS, which is then returned to the waiting parent process. The parent process then resumes execution. A child process that dies but is never waited on by its parent becomes a zombie process. Such a process continues to exist as entry in the system process table even though it is no longer an actively executing program.

Pipes

Pipes provide a unidirectional Interprocess communication channel. "|" (pipe) operator between two command directs the stdout of the first to the stdin of the second. Any of the commands may have options or arguments.

Ex: Command 1| command 2 parameter 1

ls –l | wc

For more details of these system calls

1. https://man7.org/linux/man-pages/man2/fork.2.html
2. https://man7.org/linux/man-pages/man2/waitpid.2.html
3. https://man7.org/linux/man-pages/man3/exec.3.html
4. https://man7.org/linux/man-pages/man2/pipe.2.html
5. https://linux.die.net/man/3/pipe