SSN COLLEGE OF ENGINEERING, KALAVAKKAM
(An Autonomous Institution, Affiliated to Anna University, Chennai)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

# UCS2412 - OPERATING SYSTEMS LAB

---

## Lab Exercise 1 System calls and System Commands

**Objective:**

To study the purpose and working of various system calls and system commands used in Linux

**Sample Learning Outcome:**

1. Learn about the various system calls and system commands, their purpose and options

2. Work and experience the system calls and commands

3. Understand the relation between the system calls and commands

**Best Practices:**

1. Algorithm design

2. Naming convention – for file names, variables

3. Comment usage at proper places

4. Prompt messages during reading input and displaying output

5. Error handling mechanisms for failures in system calls

6. Incremental program development

7. Modularity

8. All possible test cases in output

1. *Study the following system calls and system commands (using Linux manual pages)*

| **System Commands** | | **System calls** | |
|---|---|---|---|
| cp - -i | rmdir | fork() | opendir() |
| mv - -i | wc | execl() | readdir() |
| ls - -l | who | getpid() | open() |
| grep - -c,-v | head - -n | getppid() | read() |
| chmod | tail - -n | exit() | write() |
| cat | nl | wait() | creat() |
| mkdir | awk | close() | sleep() |
| rm | | | |

Write the following in your ***<u>observation</u>*** for each of the commands and calls.

**a) System Commands:**
Name, Purpose, options, Syntax, Example

**b) System Calls:**
Name, Description, Header file, Syntax, Arguments, Return type (both : on success and on failure)

2. *Develop a C program to understand the working of fork(), getpid(), getppid(), sleep() and wait().*
(Sample code is given in next file in LMS page for you to try)

3. *Develop a C program to create one child process using fork system call. Let the child process reads the first command line argument and compute the factorial of that number. Let the parent process reads the second command line argument and print the series upto that number.*
(Reading material is attached in the LMS page)

*Submit the code with snapshot of the output for both programs (2 & 3) one after another in a single pdf in the LMS.*

# System commands

1. Name: cp

Purpose: The cp command is used to copy files or directories.

Options:

- -r: copies directories recursively
- -i: prompts before overwriting files
- -u: copies only when the source file is newer than the destination file
- -p: preserves the file attributes, such as permissions and timestamps
- -v: displays verbose output, showing each file as it is copied

Syntax: cp [OPTIONS] source_file destination_file

Example: cp file1.txt file2.txt

2. Name: mv

Purpose: The mv command is used to move or rename files or directories.

Options:

- -i: prompts before overwriting files
- -u: moves only when the source file is newer than the destination file
- -v: displays verbose output, showing each file as it is moved

Syntax: mv [OPTIONS] source_file destination_file

Example: mv file1.txt file2.txt

3. Name: ls

Purpose: The ls command is used to list the contents of a directory.

Options:

- -l: displays the long format, showing detailed file information
- -a: shows hidden files and directories
- -h: shows file sizes in a human-readable format
- -t: sorts the files by modification time
- -r: reverses the order of the sort

Syntax: ls [OPTIONS] [FILE]...

Example: ls -l

4. Name: grep

Purpose: The grep command is used to search for a pattern in a file or input.

Options:

-i: ignores case when searching

-v: displays the lines that do not match the pattern

-c: displays the count of matches -n: displays the line numbers of matches

-r: searches recursively in directories

Syntax: grep [OPTIONS] pattern [FILE]...

Example: grep "hello" file1.txt

5. Name: chmod

Purpose: The chmod command is used to change the permissions of a file or directory.

Options:

u/g/o/a: specifies the user, group, others, or all

+/-/=: adds, removes, or sets the permissions

r/w/x: specifies read, write, or execute permissions

Syntax: chmod [OPTIONS] mode file/directory

Example: chmod u+x file1.txt

6. Name: cat

Purpose: The cat command is used to display the contents of a file.

Options:

-n: displays line numbers

-b: displays line numbers for non-blank lines

-s: squeezes consecutive blank lines into one

Syntax: cat [OPTIONS] [FILE]...

Example: cat file1.txt

7. Name: mkdir

Purpose: The mkdir command is used to create a new directory.

Options:

-p: creates parent directories if they do not exist

-v: displays verbose output, showing each directory as it is created

Syntax: mkdir [OPTIONS] directory

Example: mkdir new_directory

8. Name: rm

Purpose: The rm command is used to remove files or directories.

Options:

-i: prompts before deleting each file

-r: deletes directories and their contents recursively

-f: forces the deletion without prompting

Syntax: rm [OPTIONS] file/directory

Example: rm file1.txt

9. Name: rmdir

Purpose: The rmdir command is used to remove empty directories.

Options:

-p: removes parent directories if they become empty

-v: displays verbose output, showing each directory as it is removed

Syntax: rmdir [OPTIONS] directory

Example: rmdir empty_directory

10. Name: wc

Purpose: The wc command is used to count the number of words, lines, and characters in a file or input.

Options:

-l: displays the count of lines

-w: displays the count of words

-c: displays the count of bytes

Syntax: wc [OPTIONS] [FILE]...

Example: wc -l file1.txt

11. Name: who

Purpose: The who command is used to display information about currently logged in users.

Options:

-a: displays information about idle time and the process being run

-r: displays the current runlevel

Syntax: who [OPTIONS]

Example: who -a

12. Name: head

Purpose: The head command is used to display the first few lines of a file.

Options:

-n: specifies the number of lines to display

Syntax: head [OPTIONS] [FILE]...

Example: head -n 5 file1.txt

13. Name: tail

Purpose: The tail command is used to display the last few lines of a file.

Options:

  -n: specifies the number of lines to display

  -f: displays the lines as they are appended to the file

Syntax: tail [OPTIONS] [FILE]...

Example: tail -n 5 file1.txt

14. Name: nl

Purpose: The nl command is used to display the contents of a file with line numbers.

Options:

  -b: specifies when to number the lines

  -b a: numbers all lines

  -b t: numbers non-blank lines

  -n: specifies the line numbering format

Syntax: nl [OPTIONS] [FILE]...

Example: nl -b a file1.txt

15. Name: awk

Purpose: The awk command is used to search for and manipulate text in a file.

Options:

  -F: specifies the field separator

  -v: defines a variable

  -f: specifies a script file to execute

Syntax: awk [OPTIONS] 'pattern {action}' [FILE]...

Example: awk -F, '{print $1}' file1.txt

## System Call

**1. Name:** fork()

**Description:** The fork() system call creates a new process by duplicating the calling process.

**Header file:** <unistd.h>

**Syntax:** pid_t fork(void);

**Arguments:**None.

**Return type:**

- On success: In the parent process, the process ID of the child process is returned. In the child process, 0 is returned.
- On failure: -1 is returned and errno is set appropriately.

**2. Name:** execl()

**Description:** The execl() system call replaces the current process image with a new process image specified by the path name.

**Header file:** <unistd.h>

**Syntax:** int execl(const char *path, const char *arg, ...);

**Arguments:**

- *path: Pointer to a null-terminated string that specifies the path name of the new process image file.
- *arg: Pointer to a null-terminated string that specifies the name of the new process image.
- ...: A list of zero or more additional arguments to be passed to the new process.

**Return type:**

- On success: This function does not return, as the calling process is replaced by the new process image.
- On failure: -1 is returned and errno is set appropriately.

**3. Name:** getpid()

**Description:** The getpid() system call returns the process ID of the calling process.

**Header file:** <unistd.h>

**Syntax:** pid_t getpid(void);

**Arguments:**None.

**Return type:**

- On success: The process ID of the calling process is returned.
- On failure: -1 is returned and errno is set appropriately.

**4. Name:** getppid()

**Description:** The getppid() system call returns the parent process ID of the calling process.

**Header file:** <unistd.h>

**Syntax:** pid_t getppid(void);

**Arguments:** No arguments.

**Return type:**

- On success: The parent process ID of the calling process is returned.
- On failure: -1 is returned and errno is set appropriately.

**5. Name:** exit()

**Description:** The exit() system call terminates the calling process immediately, and returns an exit status to the parent process.

**Header file:** <stdlib.h>

**Syntax:** void exit(int status);

**Arguments:** status: An integer value that specifies the exit status of the process.

**Return type:**

- This function does not return.

**6. Name:** wait()

**Description:** The wait() system call suspends the execution of the calling process until one of its child processes terminates.

**Header file:** <sys/wait.h>

**Syntax:** pid_t wait(int *status);

**Arguments:***status: Pointer to an integer in which the exit status of the terminated child process is stored.

**Return type:**

- **On success:** The process ID of the terminated child.
- **On failure:** -1 is returned if there are no child processes, if the calling process has no children, or if the call is interrupted by a signal.

**7. Name:** close()

**Description:** The close() system call closes a file descriptor, so that it no longer refers to any file and can be reused.

**Header file:** <unistd.h>

**Syntax:** int close(int fd);

**Arguments:** fd: The file descriptor to be closed.

**Return type:**

- On success: 0 is returned.
- On failure: -1 is returned if the file descriptor is not valid or the operation is not permitted.

**8. Name:** opendir()

**Description:** The opendir() system call opens a directory stream corresponding to the directory given by the path name.

**Header file:** <dirent.h>

**Syntax:** DIR *opendir(const char *name);

**Arguments:** *name: Pointer to a null-terminated string that specifies the name of the directory.

**Return type:**

- On success: A pointer to a DIR structure representing the directory stream is returned.
- On failure: NULL is returned if the directory does not exist or if permission is denied.

**9. Name:** readdir()

**Description:** The readdir() system call reads the next directory entry from the directory stream referred to by the DIR pointer.

**Header file:** <dirent.h>

**Syntax:** struct dirent *readdir(DIR *dirp);

**Arguments:***dirp: Pointer to a DIR structure representing the directory stream.

**Return type:**

- On success: A pointer to a dirent structure containing the directory entry information is returned.
- On failure: NULL is returned at the end of the directory stream or if an error occurs.

**10. Name:** open()

**Description:** The open() system call opens a file or device and returns a file descriptor.

**Header file:** <fcntl.h>

**Syntax:** int open(const char *path, int flags, mode_t mode);

**Arguments:** *path: Pointer to a null-terminated string that specifies the file or device to be opened.

- flags: A bitwise OR of the access modes and file status flags.
- mode: A value specifying the permissions to be set for the file, in case it is created.

**Return type:**

- On success: A file descriptor is returned.
- On failure: -1 is returned and errno is set appropriately.

**11. Name:** read()

**Description:** The read() system call reads data from a file descriptor into a buffer.

**Header file:** <unistd.h>

**Syntax:** ssize_t read(int fd, void *buf, size_t count);

**Arguments:**

- fd: The file descriptor from which to read data.
- *buf: Pointer to the buffer where the data read will be stored.
- count: The maximum number of bytes to read.

**Return type:**

- On success: The number of bytes read is returned.
- On failure: -1 is returned and errno is set appropriately.

**12. Name:** write()

**Description:** The write() system call writes data to a file descriptor from a buffer.

**Header file:** <unistd.h>

**Syntax:** ssize_t write(int fd, const void *buf, size_t count);

**Arguments:**

- fd: The file descriptor to which to write data.
- *buf: Pointer to the buffer containing the data to be written.
- count: The number of bytes to be written.

**Return type:**

- On success: The number of bytes written is returned.
- On failure: -1 is returned and errno is set appropriately.

**13. Name:** creat()

**Description:** The creat() system call creates a new file or opens an existing file with write-only access.

**Header file:** <fcntl.h>

**Syntax:** int creat(const char *path, mode_t mode);

**Arguments:**

- *path: Pointer to a null-terminated string that specifies the name of the file to be created.
- mode: A value specifying the permissions to be set for the file.

**Return type:**

- On success: A file descriptor is returned.
- On failure: -1 is returned and errno is set appropriately.

**14 Name:** sleep()

**Description:** The sleep() system call suspends execution of the calling process for a specified number of seconds.

**Header file:** <unistd.h>

Syntax: unsigned int sleep(unsigned int seconds);

**Arguments:** seconds: The number of seconds to suspend execution.

**Return type:**

- On success: The number of seconds remaining to sleep is returned (0 if the sleep completed).
- On failure: The sleep function always completes successfully and returns a value of 0.

https://www.computerhope.com/jargon/v/verbose-mode.htm

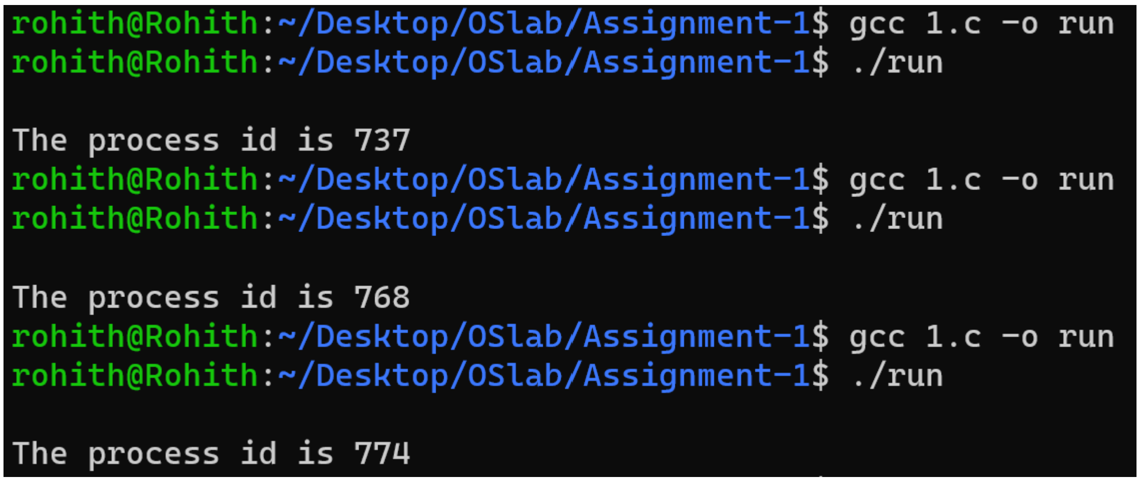
Ex-no:1 Name: M.Rohith
Date: 03-03-2023 3122 21 5001 085

## 2. **Understanding the working of getpid():**

Program code:

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

int main()
{
    pid_t ret_val;
    printf("\nThe process id is %d\n",getpid());
    return 0;
}
```

Output:



```
rohith@Rohith:~/Desktop/OSlab/Assignment-1$ gcc 1.c -o run
rohith@Rohith:~/Desktop/OSlab/Assignment-1$ ./run

The process id is 737
rohith@Rohith:~/Desktop/OSlab/Assignment-1$ gcc 1.c -o run
rohith@Rohith:~/Desktop/OSlab/Assignment-1$ ./run

The process id is 768
rohith@Rohith:~/Desktop/OSlab/Assignment-1$ gcc 1.c -o run
rohith@Rohith:~/Desktop/OSlab/Assignment-1$ ./run

The process id is 774
```

Explanation:

- The process id of the calling process will be printed by the getpid().
- Each time the program is run multiple times, unique process id is returned.

Ex-no:1 Name: M.Rohith
Date: 03-03-2023 3122 21 5001 085

## **Understanding the working of fork():**

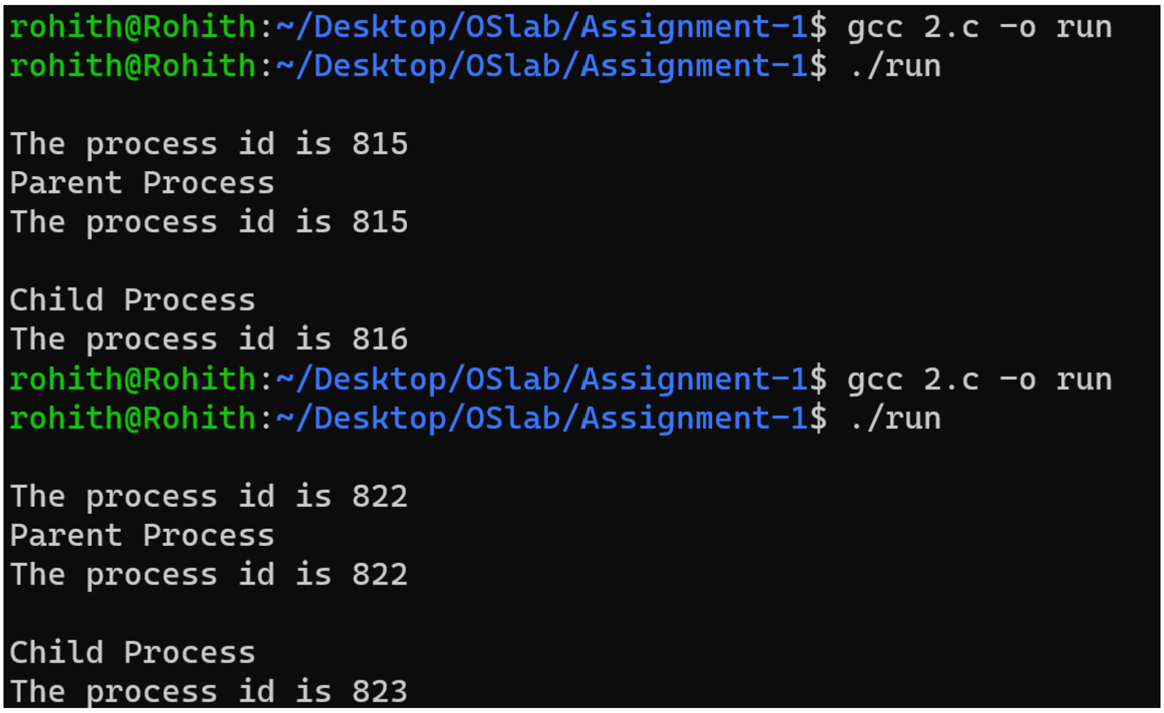
Program code:

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

int main()
{
    pid_t ret_val;
    printf("\nThe process id is %d\n",getpid());

    ret_val=fork();
    if(ret_val<0)
    {
        //fork has failed
        printf("\nFork failure\n");
    }
    else if(ret_val==0)
    {
        //child process
        printf("\nChild Process\n");
        printf("The process id is %d\n",getpid());
    }
    else
    {
        //parent process
        printf("Parent Process\n");
        printf("The process id is %d\n",getpid());
    }
    return 0;
}
```

Ex-no:1
Date: 03-03-2023

Name: M.Rohith
3122 21 5001 085

Output:



```
rohith@Rohith:~/Desktop/OSlab/Assignment-1$ gcc 2.c -o run
rohith@Rohith:~/Desktop/OSlab/Assignment-1$ ./run

The process id is 815
Parent Process
The process id is 815

Child Process
The process id is 816
rohith@Rohith:~/Desktop/OSlab/Assignment-1$ gcc 2.c -o run
rohith@Rohith:~/Desktop/OSlab/Assignment-1$ ./run

The process id is 822
Parent Process
The process id is 822

Child Process
The process id is 823
```

Explanation:

- The fork() is used to duplicate the process and after duplicating, the original process is called the parent process and the duplicate process is called as the child process.
- The fork() returns the value as pid_t
- If the return value is less than 0, then it is the failure process and if it is equal to 0, it should enter the child process and otherwise it should enter the parent process and parent process id.
- In ubuntu a prompt message will appear which means the parent process is terminated and again fork() is called which enters into the child process and prints unique process id.
- The child process will be still running, so this process becomes a defunct process also known as zombie.
- If we do not use wait() in parent process then the parent process is executed first and it terminates at the prompt message whereas the child process will still be running.

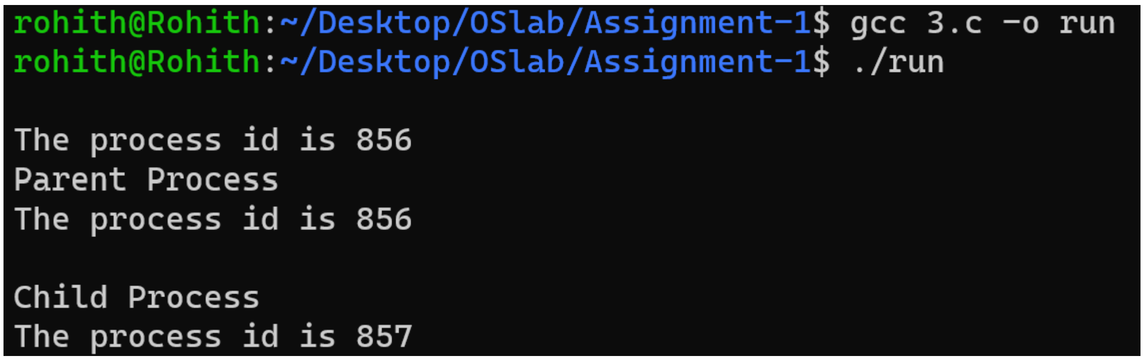
Ex-no:1 Name: M.Rohith
Date: 03-03-2023 3122 21 5001 085

## Understanding the working of sleep():

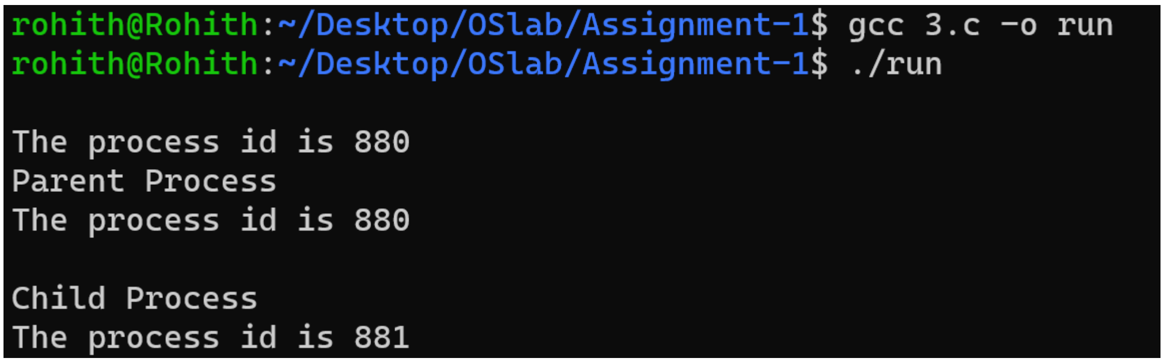
Program code:

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

int main()
{
    pid_t ret_val;
    printf("\nThe process id is %d\n",getpid());

    ret_val=fork();
    if(ret_val<0)
    {
        //fork has failed
        printf("\nFork failure\n");
    }
    else if(ret_val==0)
    {
        //child process
        printf("\nChild Process\n");
        printf("The process id is %d\n",getpid());
        sleep(20);
    }
    else
    {
        //parent process
        printf("Parent Process\n");
        printf("The process id is %d\n",getpid());
        sleep(30);
    }
    return 0;
}
```

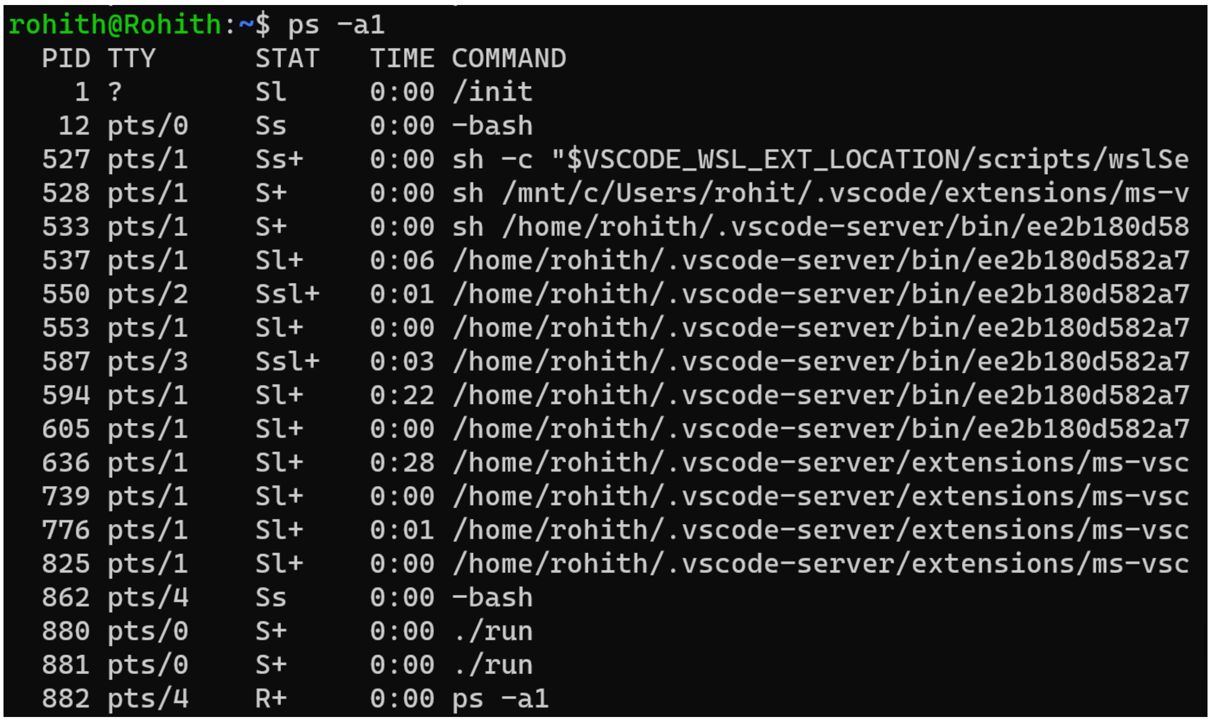
Output:



```
rohith@Rohith:~/Desktop/OSlab/Assignment-1$ gcc 3.c -o run
rohith@Rohith:~/Desktop/OSlab/Assignment-1$ ./run

The process id is 856
Parent Process
The process id is 856

Child Process
The process id is 857
```

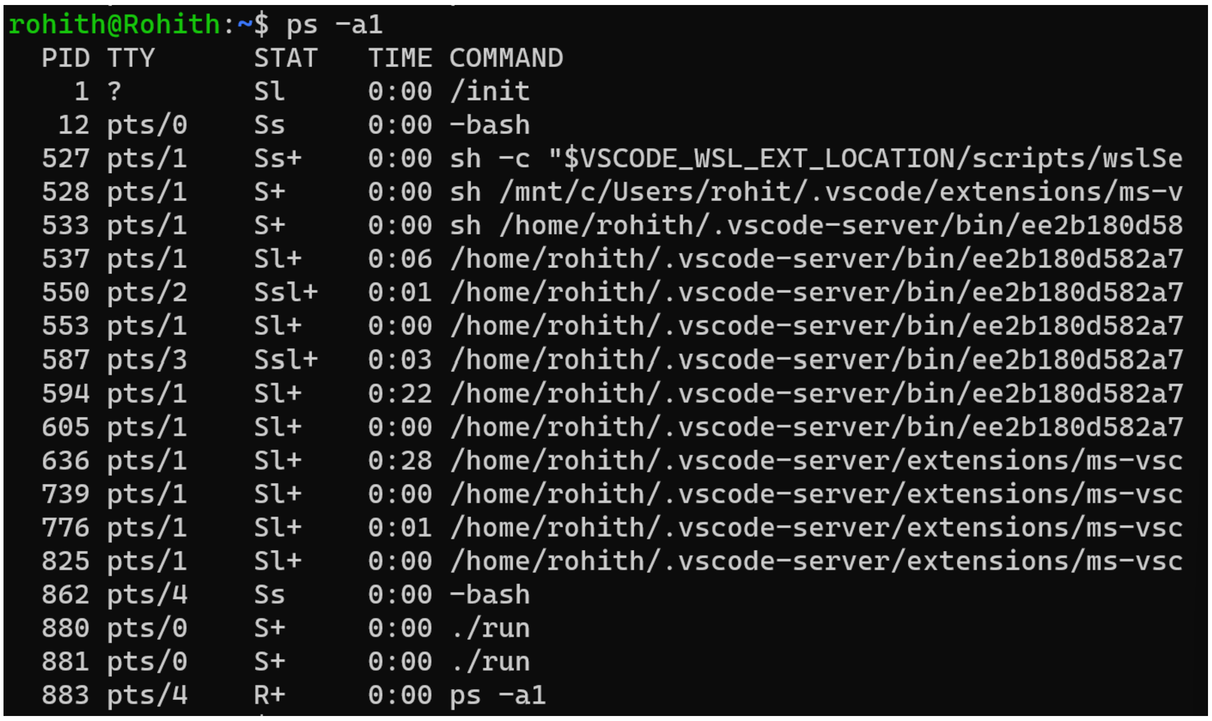


```
rohith@Rohith:~/Desktop/OSlab/Assignment-1$ gcc 3.c -o run
rohith@Rohith:~/Desktop/OSlab/Assignment-1$ ./run

The process id is 880
Parent Process
The process id is 880

Child Process
The process id is 881
```

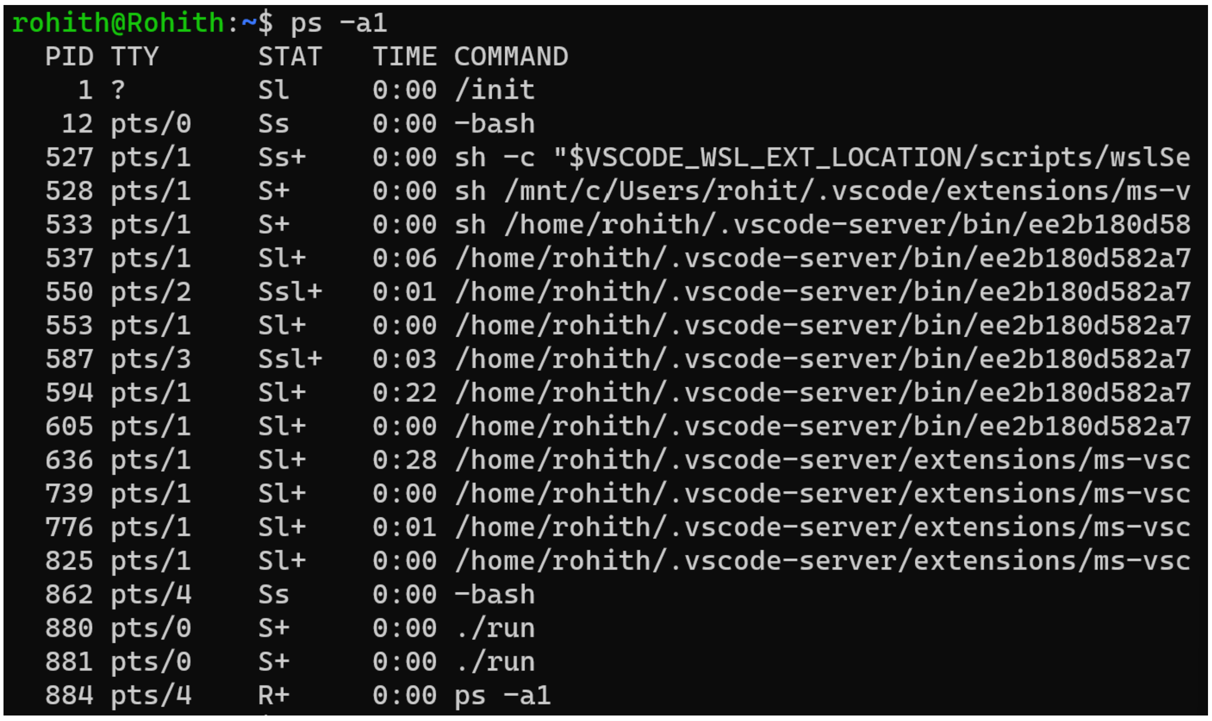


```
rohith@Rohith:~$ ps -a1
  PID TTY      STAT   TIME COMMAND
    1 ?        Sl     0:00 /init
   12 pts/0    Ss     0:00 -bash
  527 pts/1    Ss+    0:00 sh -c "$VSCODE_WSL_EXT_LOCATION/scripts/wslSe
  528 pts/1    S+     0:00 sh /mnt/c/Users/rohit/.vscode/extensions/ms-v
  533 pts/1    S+     0:00 sh /home/rohith/.vscode-server/bin/ee2b180d58
  537 pts/1    Sl+    0:06 /home/rohith/.vscode-server/bin/ee2b180d582a7
  550 pts/2    Ssl+   0:01 /home/rohith/.vscode-server/bin/ee2b180d582a7
  553 pts/1    Sl+    0:00 /home/rohith/.vscode-server/bin/ee2b180d582a7
  587 pts/3    Ssl+   0:03 /home/rohith/.vscode-server/bin/ee2b180d582a7
  594 pts/1    Sl+    0:22 /home/rohith/.vscode-server/bin/ee2b180d582a7
  605 pts/1    Sl+    0:00 /home/rohith/.vscode-server/bin/ee2b180d582a7
  636 pts/1    Sl+    0:28 /home/rohith/.vscode-server/extensions/ms-vsc
  739 pts/1    Sl+    0:00 /home/rohith/.vscode-server/extensions/ms-vsc
  776 pts/1    Sl+    0:01 /home/rohith/.vscode-server/extensions/ms-vsc
  825 pts/1    Sl+    0:00 /home/rohith/.vscode-server/extensions/ms-vsc
  862 pts/4    Ss     0:00 -bash
  880 pts/0    S+     0:00 ./run
  881 pts/0    S+     0:00 ./run
  882 pts/4    R+     0:00 ps -a1
```

Ex-no:1 Name: M.Rohith
Date: 03-03-2023 3122 21 5001 085

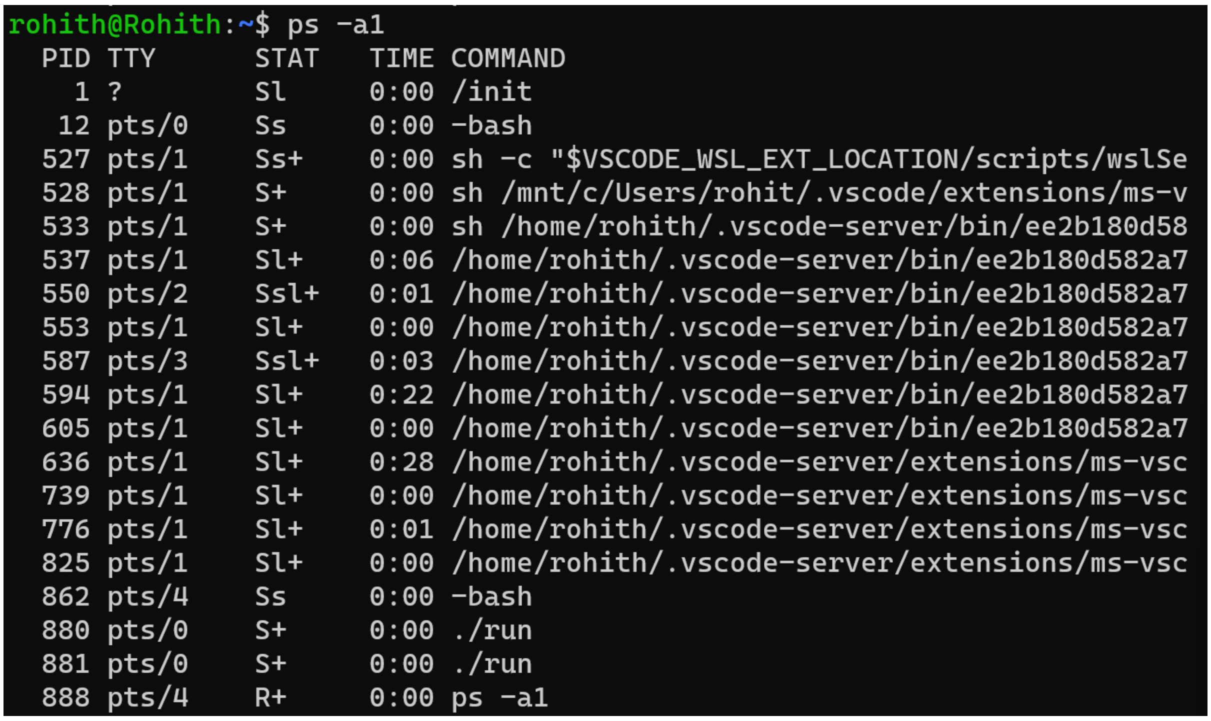


```
rohith@Rohith:~$ ps -a1
  PID TTY      STAT   TIME COMMAND
    1 ?        Sl     0:00 /init
   12 pts/0    Ss     0:00 -bash
  527 pts/1    Ss+    0:00 sh -c "$VSCODE_WSL_EXT_LOCATION/scripts/wslSe
  528 pts/1    S+     0:00 sh /mnt/c/Users/rohit/.vscode/extensions/ms-v
  533 pts/1    S+     0:00 sh /home/rohith/.vscode-server/bin/ee2b180d58
  537 pts/1    Sl+    0:06 /home/rohith/.vscode-server/bin/ee2b180d582a7
  550 pts/2    Ssl+   0:01 /home/rohith/.vscode-server/bin/ee2b180d582a7
  553 pts/1    Sl+    0:00 /home/rohith/.vscode-server/bin/ee2b180d582a7
  587 pts/3    Ssl+   0:03 /home/rohith/.vscode-server/bin/ee2b180d582a7
  594 pts/1    Sl+    0:22 /home/rohith/.vscode-server/bin/ee2b180d582a7
  605 pts/1    Sl+    0:00 /home/rohith/.vscode-server/bin/ee2b180d582a7
  636 pts/1    Sl+    0:28 /home/rohith/.vscode-server/extensions/ms-vsc
  739 pts/1    Sl+    0:00 /home/rohith/.vscode-server/extensions/ms-vsc
  776 pts/1    Sl+    0:01 /home/rohith/.vscode-server/extensions/ms-vsc
  825 pts/1    Sl+    0:00 /home/rohith/.vscode-server/extensions/ms-vsc
  862 pts/4    Ss     0:00 -bash
  880 pts/0    S+     0:00 ./run
  881 pts/0    S+     0:00 ./run
  883 pts/4    R+     0:00 ps -a1
```

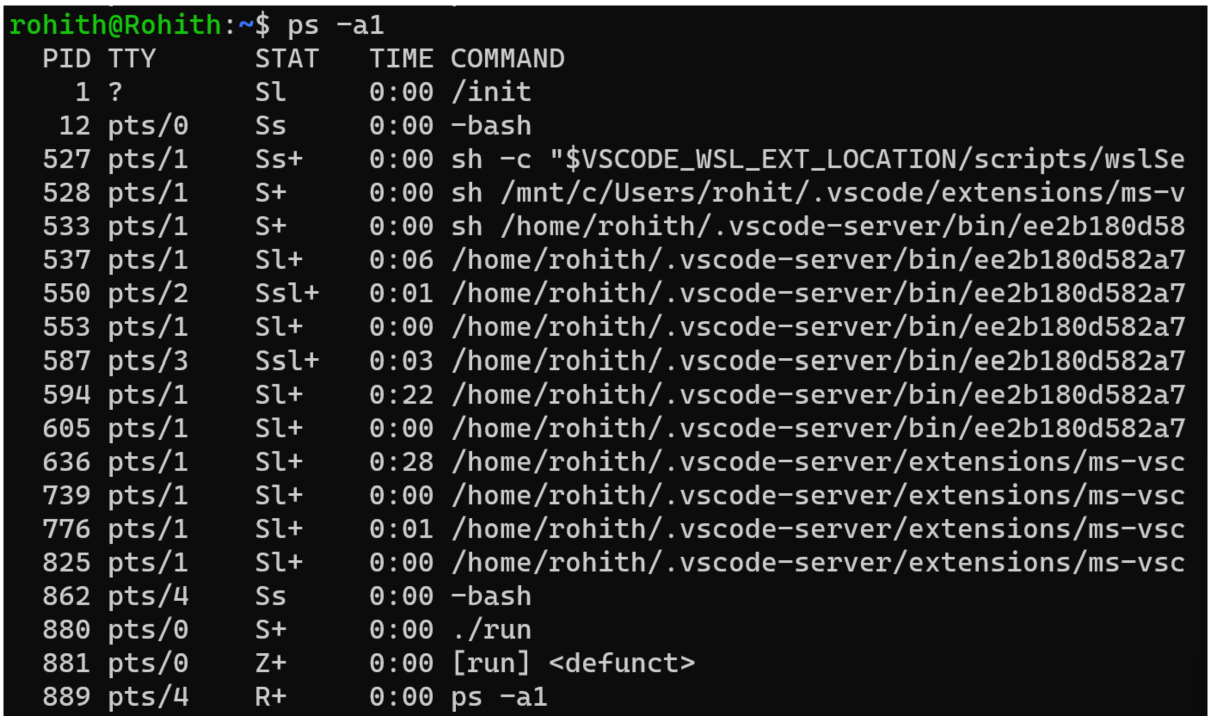


```
rohith@Rohith:~$ ps -a1
  PID TTY      STAT   TIME COMMAND
    1 ?        Sl     0:00 /init
   12 pts/0    Ss     0:00 -bash
  527 pts/1    Ss+    0:00 sh -c "$VSCODE_WSL_EXT_LOCATION/scripts/wslSe
  528 pts/1    S+     0:00 sh /mnt/c/Users/rohit/.vscode/extensions/ms-v
  533 pts/1    S+     0:00 sh /home/rohith/.vscode-server/bin/ee2b180d58
  537 pts/1    Sl+    0:06 /home/rohith/.vscode-server/bin/ee2b180d582a7
  550 pts/2    Ssl+   0:01 /home/rohith/.vscode-server/bin/ee2b180d582a7
  553 pts/1    Sl+    0:00 /home/rohith/.vscode-server/bin/ee2b180d582a7
  587 pts/3    Ssl+   0:03 /home/rohith/.vscode-server/bin/ee2b180d582a7
  594 pts/1    Sl+    0:22 /home/rohith/.vscode-server/bin/ee2b180d582a7
  605 pts/1    Sl+    0:00 /home/rohith/.vscode-server/bin/ee2b180d582a7
  636 pts/1    Sl+    0:28 /home/rohith/.vscode-server/extensions/ms-vsc
  739 pts/1    Sl+    0:00 /home/rohith/.vscode-server/extensions/ms-vsc
  776 pts/1    Sl+    0:01 /home/rohith/.vscode-server/extensions/ms-vsc
  825 pts/1    Sl+    0:00 /home/rohith/.vscode-server/extensions/ms-vsc
  862 pts/4    Ss     0:00 -bash
  880 pts/0    S+     0:00 ./run
  881 pts/0    S+     0:00 ./run
  884 pts/4    R+     0:00 ps -a1
```

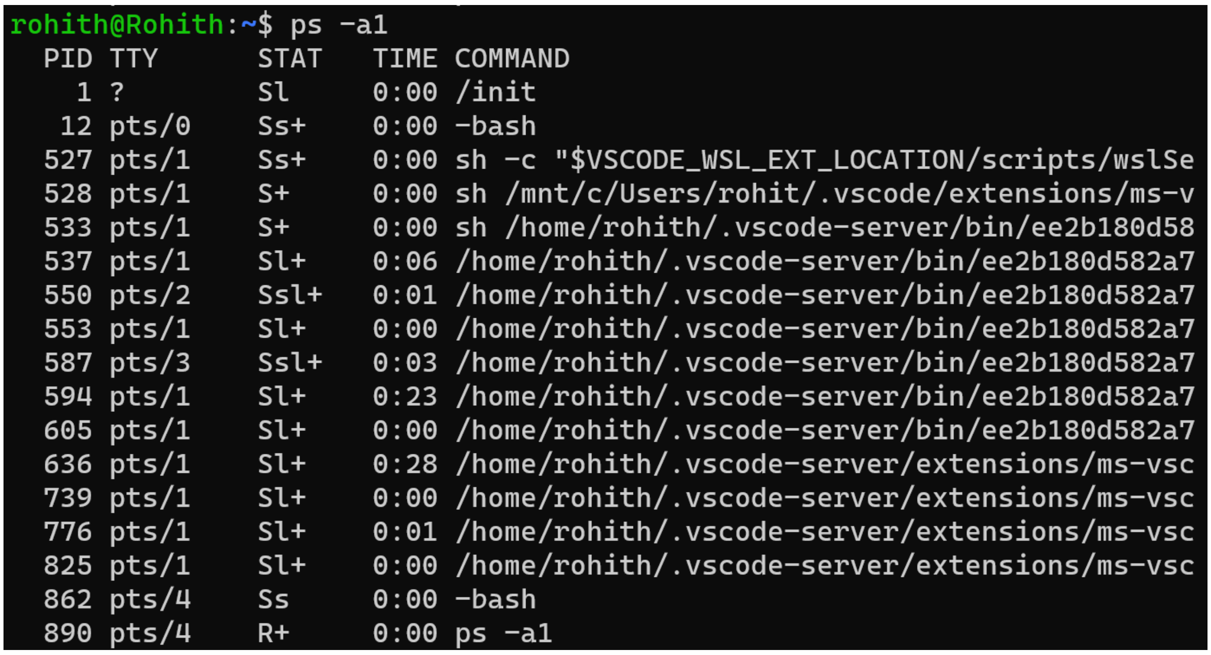
Ex-no:1 Name: M.Rohith
Date: 03-03-2023 3122 21 5001 085



```
rohith@Rohith:~$ ps -a1
  PID TTY      STAT   TIME COMMAND
    1 ?        Sl     0:00 /init
   12 pts/0    Ss     0:00 -bash
  527 pts/1    Ss+    0:00 sh -c "$VSCODE_WSL_EXT_LOCATION/scripts/wslSe
  528 pts/1    S+     0:00 sh /mnt/c/Users/rohit/.vscode/extensions/ms-v
  533 pts/1    S+     0:00 sh /home/rohith/.vscode-server/bin/ee2b180d58
  537 pts/1    Sl+    0:06 /home/rohith/.vscode-server/bin/ee2b180d582a7
  550 pts/2    Ssl+   0:01 /home/rohith/.vscode-server/bin/ee2b180d582a7
  553 pts/1    Sl+    0:00 /home/rohith/.vscode-server/bin/ee2b180d582a7
  587 pts/3    Ssl+   0:03 /home/rohith/.vscode-server/bin/ee2b180d582a7
  594 pts/1    Sl+    0:22 /home/rohith/.vscode-server/bin/ee2b180d582a7
  605 pts/1    Sl+    0:00 /home/rohith/.vscode-server/bin/ee2b180d582a7
  636 pts/1    Sl+    0:28 /home/rohith/.vscode-server/extensions/ms-vsc
  739 pts/1    Sl+    0:00 /home/rohith/.vscode-server/extensions/ms-vsc
  776 pts/1    Sl+    0:01 /home/rohith/.vscode-server/extensions/ms-vsc
  825 pts/1    Sl+    0:00 /home/rohith/.vscode-server/extensions/ms-vsc
  862 pts/4    Ss     0:00 -bash
  880 pts/0    S+     0:00 ./run
  881 pts/0    S+     0:00 ./run
  888 pts/4    R+     0:00 ps -a1
```



```
rohith@Rohith:~$ ps -a1
  PID TTY      STAT   TIME COMMAND
    1 ?        Sl     0:00 /init
   12 pts/0    Ss     0:00 -bash
  527 pts/1    Ss+    0:00 sh -c "$VSCODE_WSL_EXT_LOCATION/scripts/wslSe
  528 pts/1    S+     0:00 sh /mnt/c/Users/rohit/.vscode/extensions/ms-v
  533 pts/1    S+     0:00 sh /home/rohith/.vscode-server/bin/ee2b180d58
  537 pts/1    Sl+    0:06 /home/rohith/.vscode-server/bin/ee2b180d582a7
  550 pts/2    Ssl+   0:01 /home/rohith/.vscode-server/bin/ee2b180d582a7
  553 pts/1    Sl+    0:00 /home/rohith/.vscode-server/bin/ee2b180d582a7
  587 pts/3    Ssl+   0:03 /home/rohith/.vscode-server/bin/ee2b180d582a7
  594 pts/1    Sl+    0:22 /home/rohith/.vscode-server/bin/ee2b180d582a7
  605 pts/1    Sl+    0:00 /home/rohith/.vscode-server/bin/ee2b180d582a7
  636 pts/1    Sl+    0:28 /home/rohith/.vscode-server/extensions/ms-vsc
  739 pts/1    Sl+    0:00 /home/rohith/.vscode-server/extensions/ms-vsc
  776 pts/1    Sl+    0:01 /home/rohith/.vscode-server/extensions/ms-vsc
  825 pts/1    Sl+    0:00 /home/rohith/.vscode-server/extensions/ms-vsc
  862 pts/4    Ss     0:00 -bash
  880 pts/0    S+     0:00 ./run
  881 pts/0    Z+     0:00 [run] <defunct>
  889 pts/4    R+     0:00 ps -a1
```



```
rohith@Rohith:~$ ps -a1
  PID TTY      STAT   TIME COMMAND
    1 ?        Sl     0:00 /init
   12 pts/0    Ss+    0:00 -bash
  527 pts/1    Ss+    0:00 sh -c "$VSCODE_WSL_EXT_LOCATION/scripts/wslSe
  528 pts/1    S+     0:00 sh /mnt/c/Users/rohit/.vscode/extensions/ms-v
  533 pts/1    S+     0:00 sh /home/rohith/.vscode-server/bin/ee2b180d58
  537 pts/1    Sl+    0:06 /home/rohith/.vscode-server/bin/ee2b180d582a7
  550 pts/2    Ssl+   0:01 /home/rohith/.vscode-server/bin/ee2b180d582a7
  553 pts/1    Sl+    0:00 /home/rohith/.vscode-server/bin/ee2b180d582a7
  587 pts/3    Ssl+   0:03 /home/rohith/.vscode-server/bin/ee2b180d582a7
  594 pts/1    Sl+    0:23 /home/rohith/.vscode-server/bin/ee2b180d582a7
  605 pts/1    Sl+    0:00 /home/rohith/.vscode-server/bin/ee2b180d582a7
  636 pts/1    Sl+    0:28 /home/rohith/.vscode-server/extensions/ms-vsc
  739 pts/1    Sl+    0:00 /home/rohith/.vscode-server/extensions/ms-vsc
  776 pts/1    Sl+    0:01 /home/rohith/.vscode-server/extensions/ms-vsc
  825 pts/1    Sl+    0:00 /home/rohith/.vscode-server/extensions/ms-vsc
  862 pts/4    Ss     0:00 -bash
  890 pts/4    R+     0:00 ps -a1
```

Explanation:

- More delay is added in the parent process.
- In a new terminal, to check the status of ps -a1 is used. This command is used to list the various process along with PID, TTY-terminal at which it is running and STAT .
- The parent process id 880 and child process id 881 are still running, after running two or three times, the child process id 881 will become the defunct process and now the child process will be terminated
- Once again when ps -a1 command is executed, both the process id 880 and 881 will be terminated and hence will not be found in the list.

Ex-no:1 Name: M.Rohith
Date: 03-03-2023 3122 21 5001 085

## Understanding the working of wait():

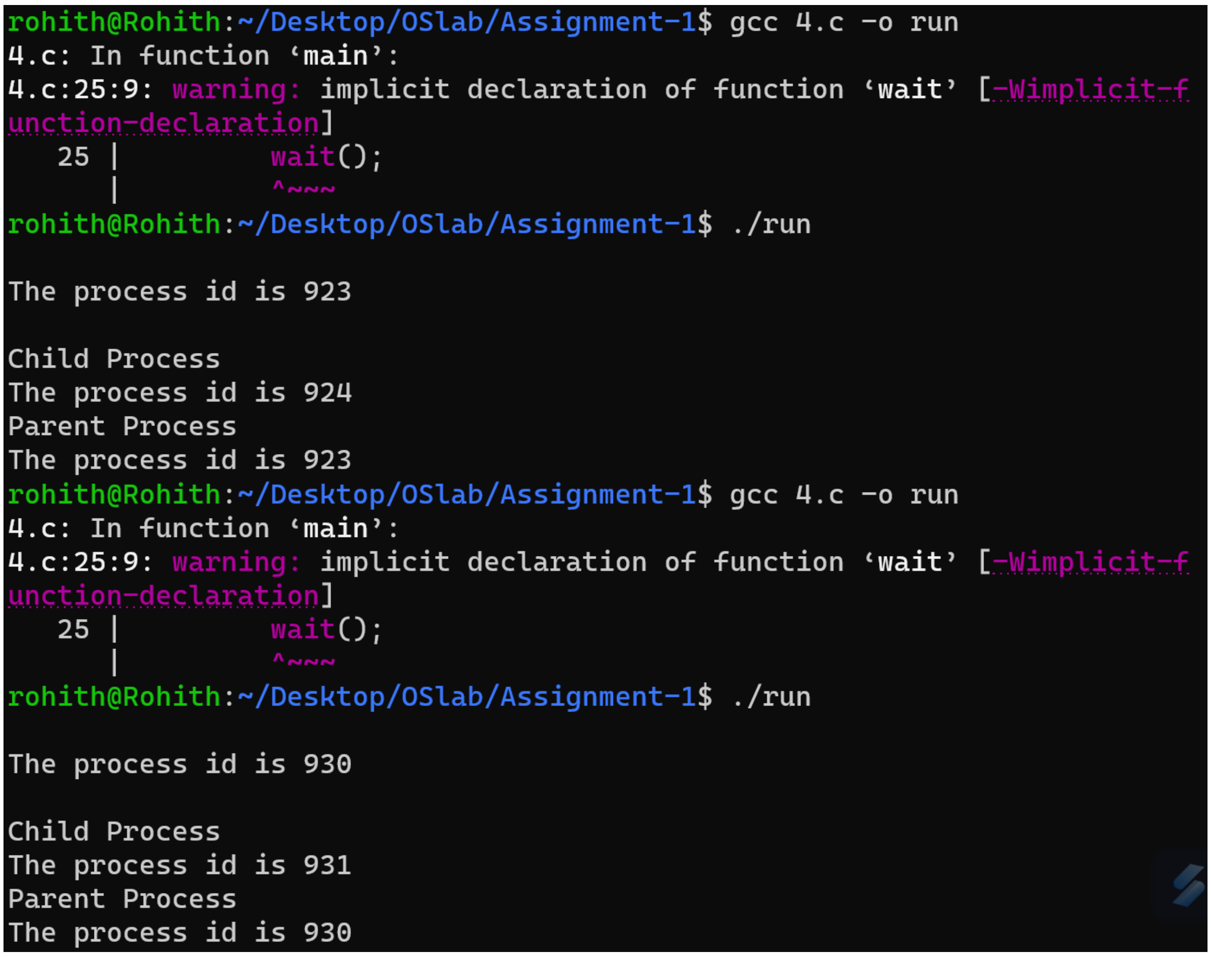
Program code:

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

int main()
{
    pid_t ret_val;
    printf("\nThe process id is %d\n",getpid());

    ret_val=fork();
    if(ret_val<0)
    {
        //fork has failed
        printf("\nFork failure\n");
    }
    else if(ret_val==0)
    {
        //child process
        printf("\nChild Process\n");
        printf("The process id is %d\n",getpid());
        sleep(20);
    }
    else
    {
        //parent process
        wait();
        printf("Parent Process\n");
        printf("The process id is %d\n",getpid());
        sleep(30);
    }
    return 0;
}
```

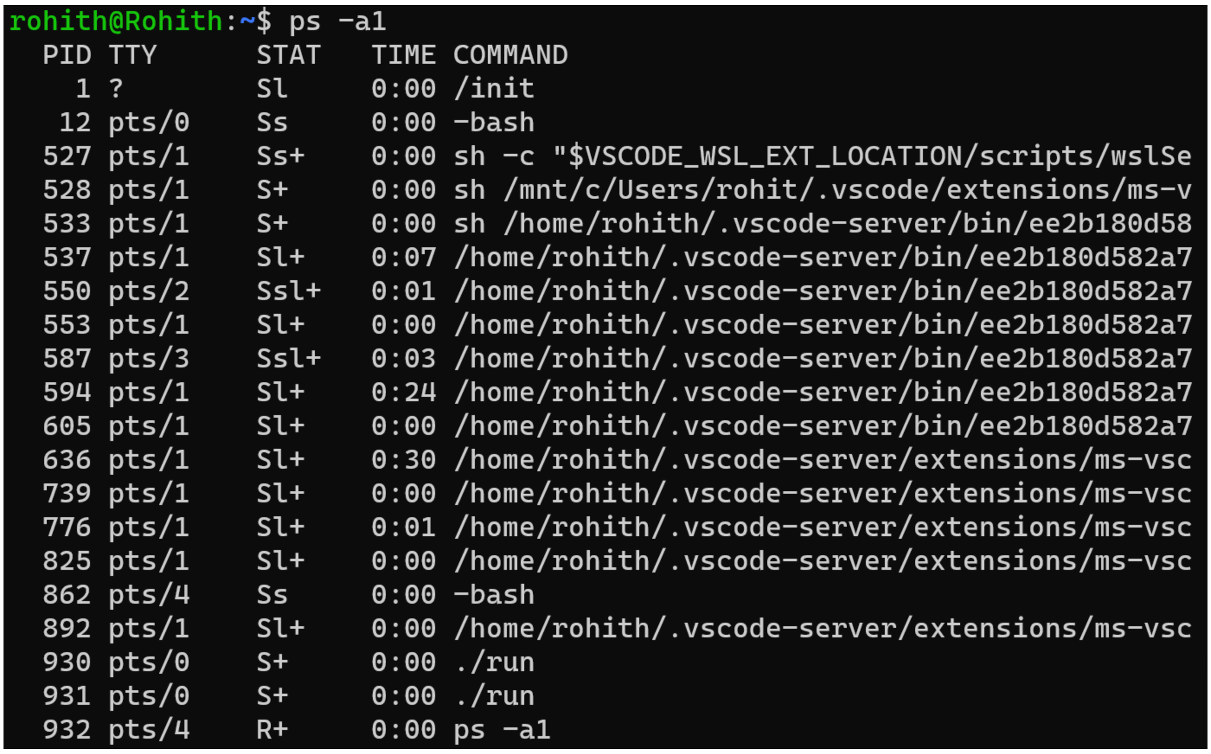
Ex-no:1
Date: 03-03-2023

Name: M.Rohith
3122 21 5001 085

Output:

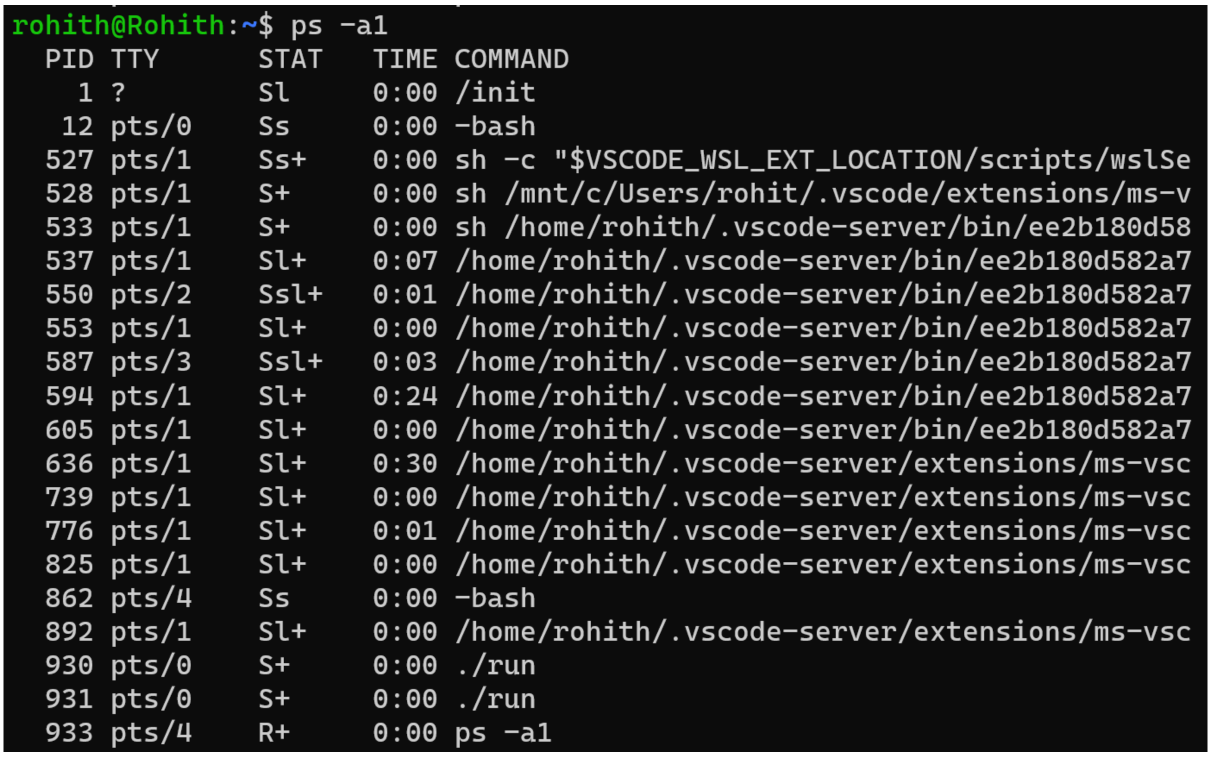


```
rohith@Rohith:~/Desktop/OSlab/Assignment-1$ gcc 4.c -o run
4.c: In function ‘main’:
4.c:25:9: warning: implicit declaration of function ‘wait’ [-Wimplicit-f
unction-declaration]
   25 |         wait();
      |         ^~~~
rohith@Rohith:~/Desktop/OSlab/Assignment-1$ ./run

The process id is 923

Child Process
The process id is 924
Parent Process
The process id is 923
rohith@Rohith:~/Desktop/OSlab/Assignment-1$ gcc 4.c -o run
4.c: In function ‘main’:
4.c:25:9: warning: implicit declaration of function ‘wait’ [-Wimplicit-f
unction-declaration]
   25 |         wait();
      |         ^~~~
rohith@Rohith:~/Desktop/OSlab/Assignment-1$ ./run

The process id is 930

Child Process
The process id is 931
Parent Process
The process id is 930
```

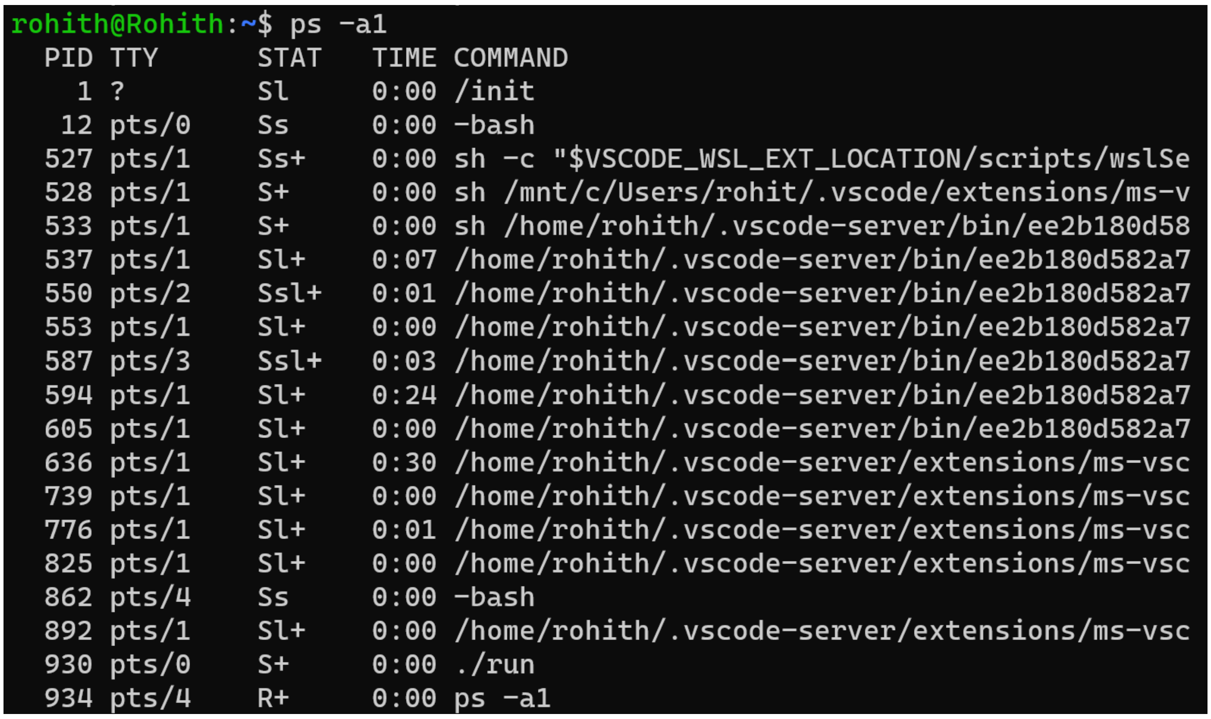


```
rohith@Rohith:~$ ps -a1
  PID TTY      STAT   TIME COMMAND
    1 ?        Sl     0:00 /init
   12 pts/0    Ss     0:00 -bash
  527 pts/1    Ss+    0:00 sh -c "$VSCODE_WSL_EXT_LOCATION/scripts/wslSe
  528 pts/1    S+     0:00 sh /mnt/c/Users/rohit/.vscode/extensions/ms-v
  533 pts/1    S+     0:00 sh /home/rohith/.vscode-server/bin/ee2b180d58
  537 pts/1    Sl+    0:07 /home/rohith/.vscode-server/bin/ee2b180d582a7
  550 pts/2    Ssl+   0:01 /home/rohith/.vscode-server/bin/ee2b180d582a7
  553 pts/1    Sl+    0:00 /home/rohith/.vscode-server/bin/ee2b180d582a7
  587 pts/3    Ssl+   0:03 /home/rohith/.vscode-server/bin/ee2b180d582a7
  594 pts/1    Sl+    0:24 /home/rohith/.vscode-server/bin/ee2b180d582a7
  605 pts/1    Sl+    0:00 /home/rohith/.vscode-server/bin/ee2b180d582a7
  636 pts/1    Sl+    0:30 /home/rohith/.vscode-server/extensions/ms-vsc
  739 pts/1    Sl+    0:00 /home/rohith/.vscode-server/extensions/ms-vsc
  776 pts/1    Sl+    0:01 /home/rohith/.vscode-server/extensions/ms-vsc
  825 pts/1    Sl+    0:00 /home/rohith/.vscode-server/extensions/ms-vsc
  862 pts/4    Ss     0:00 -bash
  892 pts/1    Sl+    0:00 /home/rohith/.vscode-server/extensions/ms-vsc
  930 pts/0    S+     0:00 ./run
  931 pts/0    S+     0:00 ./run
  932 pts/4    R+     0:00 ps -a1
```

Ex-no:1 Name: M.Rohith
Date: 03-03-2023 3122 21 5001 085

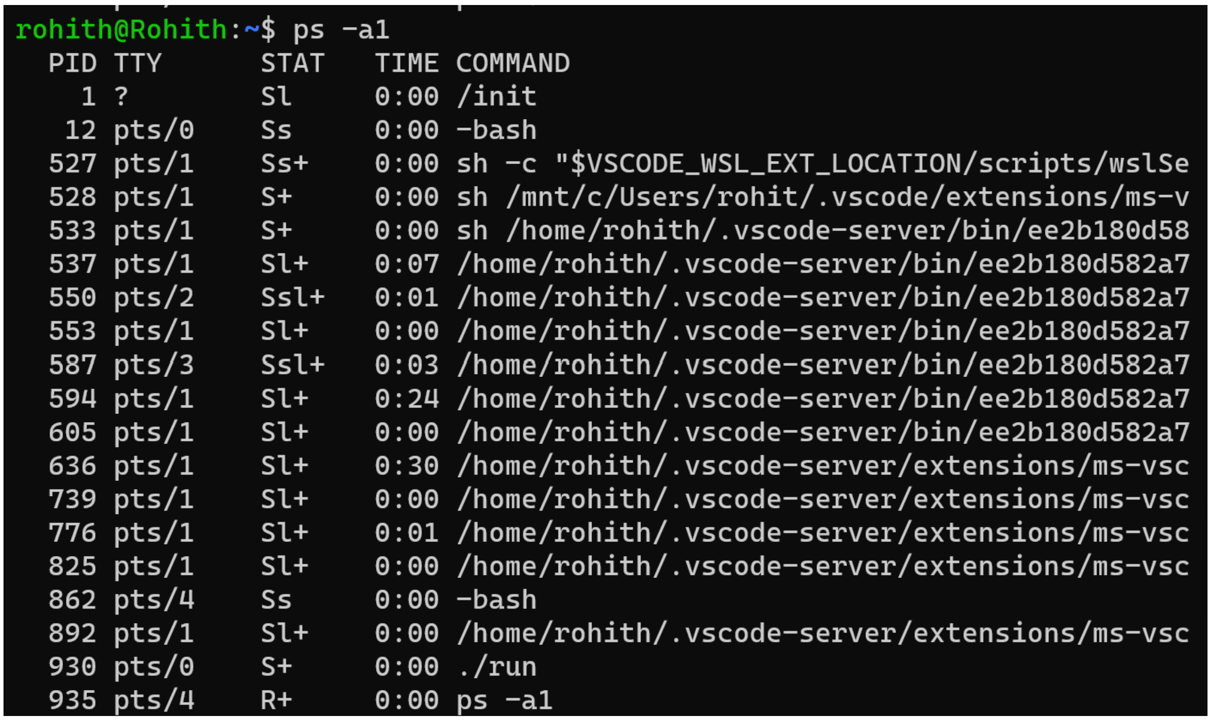


```
rohith@Rohith:~$ ps -a1
  PID TTY      STAT   TIME COMMAND
    1 ?        Sl     0:00 /init
   12 pts/0    Ss     0:00 -bash
  527 pts/1    Ss+    0:00 sh -c "$VSCODE_WSL_EXT_LOCATION/scripts/wslSe
  528 pts/1    S+     0:00 sh /mnt/c/Users/rohit/.vscode/extensions/ms-v
  533 pts/1    S+     0:00 sh /home/rohith/.vscode-server/bin/ee2b180d58
  537 pts/1    Sl+    0:07 /home/rohith/.vscode-server/bin/ee2b180d582a7
  550 pts/2    Ssl+   0:01 /home/rohith/.vscode-server/bin/ee2b180d582a7
  553 pts/1    Sl+    0:00 /home/rohith/.vscode-server/bin/ee2b180d582a7
  587 pts/3    Ssl+   0:03 /home/rohith/.vscode-server/bin/ee2b180d582a7
  594 pts/1    Sl+    0:24 /home/rohith/.vscode-server/bin/ee2b180d582a7
  605 pts/1    Sl+    0:00 /home/rohith/.vscode-server/bin/ee2b180d582a7
  636 pts/1    Sl+    0:30 /home/rohith/.vscode-server/extensions/ms-vsc
  739 pts/1    Sl+    0:00 /home/rohith/.vscode-server/extensions/ms-vsc
  776 pts/1    Sl+    0:01 /home/rohith/.vscode-server/extensions/ms-vsc
  825 pts/1    Sl+    0:00 /home/rohith/.vscode-server/extensions/ms-vsc
  862 pts/4    Ss     0:00 -bash
  892 pts/1    Sl+    0:00 /home/rohith/.vscode-server/extensions/ms-vsc
  930 pts/0    S+     0:00 ./run
  931 pts/0    S+     0:00 ./run
  933 pts/4    R+     0:00 ps -a1
```

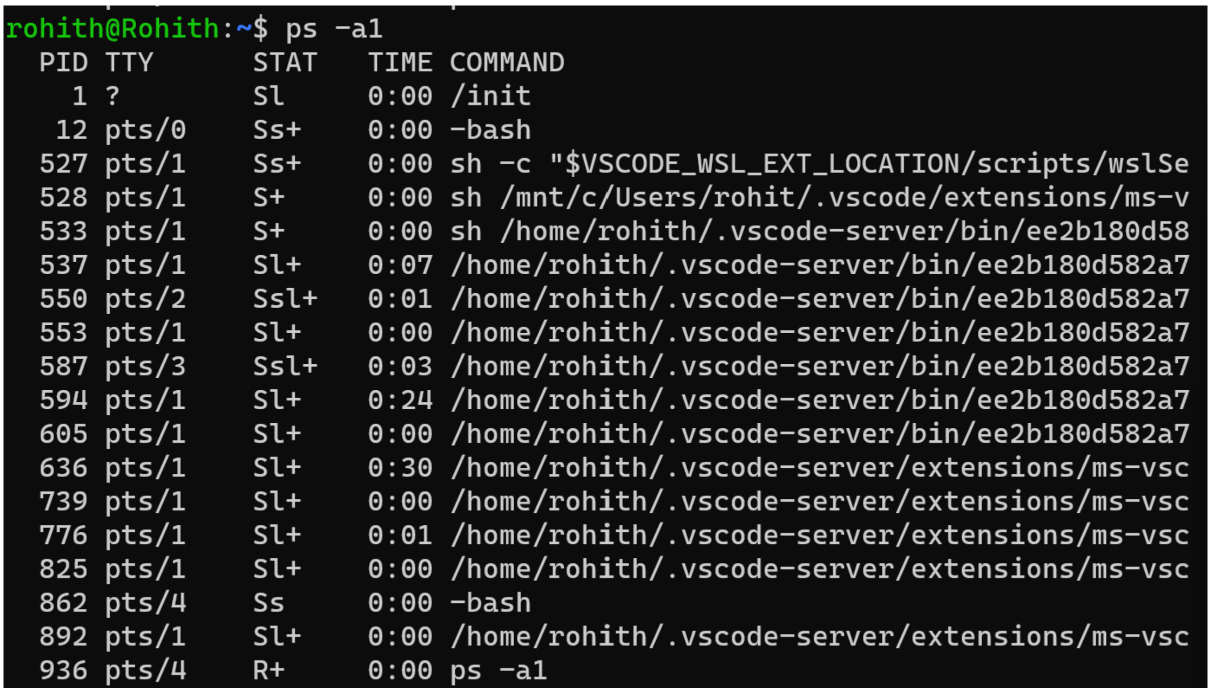


```
rohith@Rohith:~$ ps -a1
  PID TTY      STAT   TIME COMMAND
    1 ?        Sl     0:00 /init
   12 pts/0    Ss     0:00 -bash
  527 pts/1    Ss+    0:00 sh -c "$VSCODE_WSL_EXT_LOCATION/scripts/wslSe
  528 pts/1    S+     0:00 sh /mnt/c/Users/rohit/.vscode/extensions/ms-v
  533 pts/1    S+     0:00 sh /home/rohith/.vscode-server/bin/ee2b180d58
  537 pts/1    Sl+    0:07 /home/rohith/.vscode-server/bin/ee2b180d582a7
  550 pts/2    Ssl+   0:01 /home/rohith/.vscode-server/bin/ee2b180d582a7
  553 pts/1    Sl+    0:00 /home/rohith/.vscode-server/bin/ee2b180d582a7
  587 pts/3    Ssl+   0:03 /home/rohith/.vscode-server/bin/ee2b180d582a7
  594 pts/1    Sl+    0:24 /home/rohith/.vscode-server/bin/ee2b180d582a7
  605 pts/1    Sl+    0:00 /home/rohith/.vscode-server/bin/ee2b180d582a7
  636 pts/1    Sl+    0:30 /home/rohith/.vscode-server/extensions/ms-vsc
  739 pts/1    Sl+    0:00 /home/rohith/.vscode-server/extensions/ms-vsc
  776 pts/1    Sl+    0:01 /home/rohith/.vscode-server/extensions/ms-vsc
  825 pts/1    Sl+    0:00 /home/rohith/.vscode-server/extensions/ms-vsc
  862 pts/4    Ss     0:00 -bash
  892 pts/1    Sl+    0:00 /home/rohith/.vscode-server/extensions/ms-vsc
  930 pts/0    S+     0:00 ./run
  934 pts/4    R+     0:00 ps -a1
```

Ex-no:1 Name: M.Rohith
Date: 03-03-2023 3122 21 5001 085



```
rohith@Rohith:~$ ps -a1
  PID TTY      STAT   TIME COMMAND
    1 ?        Sl     0:00 /init
   12 pts/0    Ss     0:00 -bash
  527 pts/1    Ss+    0:00 sh -c "$VSCODE_WSL_EXT_LOCATION/scripts/wslSe
  528 pts/1    S+     0:00 sh /mnt/c/Users/rohit/.vscode/extensions/ms-v
  533 pts/1    S+     0:00 sh /home/rohith/.vscode-server/bin/ee2b180d58
  537 pts/1    Sl+    0:07 /home/rohith/.vscode-server/bin/ee2b180d582a7
  550 pts/2    Ssl+   0:01 /home/rohith/.vscode-server/bin/ee2b180d582a7
  553 pts/1    Sl+    0:00 /home/rohith/.vscode-server/bin/ee2b180d582a7
  587 pts/3    Ssl+   0:03 /home/rohith/.vscode-server/bin/ee2b180d582a7
  594 pts/1    Sl+    0:24 /home/rohith/.vscode-server/bin/ee2b180d582a7
  605 pts/1    Sl+    0:00 /home/rohith/.vscode-server/bin/ee2b180d582a7
  636 pts/1    Sl+    0:30 /home/rohith/.vscode-server/extensions/ms-vsc
  739 pts/1    Sl+    0:00 /home/rohith/.vscode-server/extensions/ms-vsc
  776 pts/1    Sl+    0:01 /home/rohith/.vscode-server/extensions/ms-vsc
  825 pts/1    Sl+    0:00 /home/rohith/.vscode-server/extensions/ms-vsc
  862 pts/4    Ss     0:00 -bash
  892 pts/1    Sl+    0:00 /home/rohith/.vscode-server/extensions/ms-vsc
  930 pts/0    S+     0:00 ./run
  935 pts/4    R+     0:00 ps -a1
```



```
rohith@Rohith:~$ ps -a1
  PID TTY      STAT   TIME COMMAND
    1 ?        Sl     0:00 /init
   12 pts/0    Ss+    0:00 -bash
  527 pts/1    Ss+    0:00 sh -c "$VSCODE_WSL_EXT_LOCATION/scripts/wslSe
  528 pts/1    S+     0:00 sh /mnt/c/Users/rohit/.vscode/extensions/ms-v
  533 pts/1    S+     0:00 sh /home/rohith/.vscode-server/bin/ee2b180d58
  537 pts/1    Sl+    0:07 /home/rohith/.vscode-server/bin/ee2b180d582a7
  550 pts/2    Ssl+   0:01 /home/rohith/.vscode-server/bin/ee2b180d582a7
  553 pts/1    Sl+    0:00 /home/rohith/.vscode-server/bin/ee2b180d582a7
  587 pts/3    Ssl+   0:03 /home/rohith/.vscode-server/bin/ee2b180d582a7
  594 pts/1    Sl+    0:24 /home/rohith/.vscode-server/bin/ee2b180d582a7
  605 pts/1    Sl+    0:00 /home/rohith/.vscode-server/bin/ee2b180d582a7
  636 pts/1    Sl+    0:30 /home/rohith/.vscode-server/extensions/ms-vsc
  739 pts/1    Sl+    0:00 /home/rohith/.vscode-server/extensions/ms-vsc
  776 pts/1    Sl+    0:01 /home/rohith/.vscode-server/extensions/ms-vsc
  825 pts/1    Sl+    0:00 /home/rohith/.vscode-server/extensions/ms-vsc
  862 pts/4    Ss     0:00 -bash
  892 pts/1    Sl+    0:00 /home/rohith/.vscode-server/extensions/ms-vsc
  936 pts/4    R+     0:00 ps -a1
```

Explanation:

- In order to avoid the process become a zombie, wait() is used in the parent process.
- When the above code is executed, then the parent process will be called only after the child process is terminated.

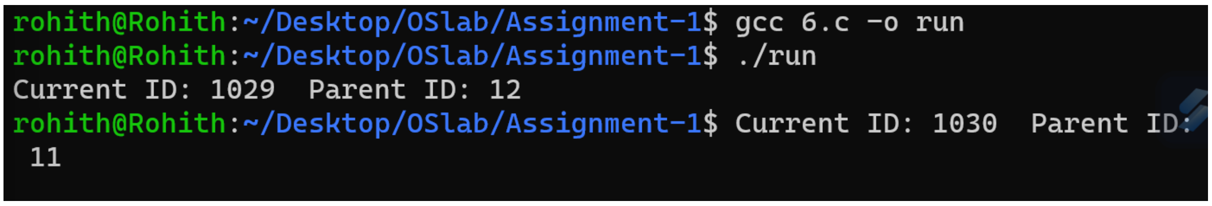
- So when ps -a1 is executed then only the parent process id 930 alone will be running as the child process id gets terminated.
- When both the parent and child process gets terminated, then when ps -a1 is executed then process id 930 and 931 will not be listed.

**Understanding the working of getppid():**

Program code:

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

int main()
{
    int id=fork();
    if(id==0)
    {
        sleep(10);
    }
    printf("Current ID: %d  Parent ID: %d\n",getpid(),getppid());
    return 0;
}
```

Output:



```
rohith@Rohith:~/Desktop/OSlab/Assignment-1$ gcc 6.c -o run
rohith@Rohith:~/Desktop/OSlab/Assignment-1$ ./run
Current ID: 1029  Parent ID: 12
rohith@Rohith:~/Desktop/OSlab/Assignment-1$ Current ID: 1030  Parent ID:
 11
```

Explanation:

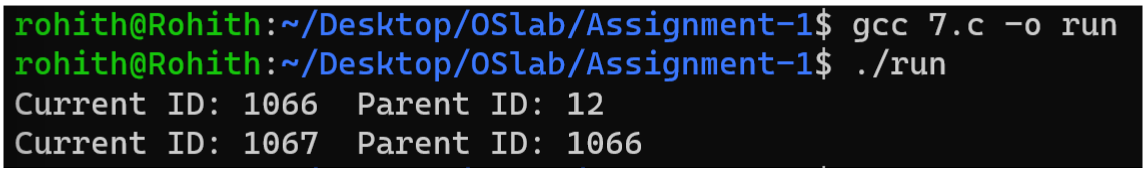
- getppid() means get parent process id.
- Every process in the system has the parent process id.

- Always the child process should terminate before the parent process. Id is 0 for child process.
- In the above program, the sleep(10) will make the child process to wait for the parent process to finish and if the parent process gets terminated before the child process then new parent process id will be assigned to that child process.

Program code:

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

int main()
{
    int id=fork();
    if(id==0)
    {
        sleep(10);
    }
    printf("Current ID: %d  Parent ID: %d\n",getpid(),getppid());
    wait(NULL); //wait for the child process to complete
    return 0;
}
```

Output:



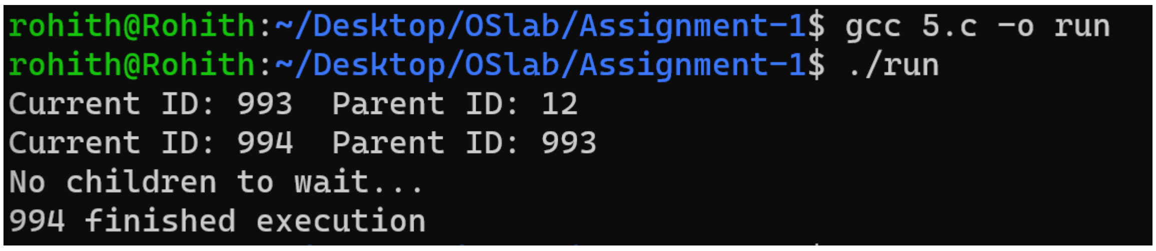
```
rohith@Rohith:~/Desktop/OSlab/Assignment-1$ gcc 7.c -o run
rohith@Rohith:~/Desktop/OSlab/Assignment-1$ ./run
Current ID: 1066   Parent ID: 12
Current ID: 1067   Parent ID: 1066
```

Program code:

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

```c
int main()
{
    int id=fork();
    if(id==0)
    {
        sleep(10);
    }
    printf("Current ID: %d  Parent ID: %d\n",getpid(),getppid());

    int res=wait(NULL); //wait for the child to die
    if(res==-1)
    {
        printf("No children to wait...\n");
    }
    else
    {
        printf("%d finished execution\n",res);
    }
}
```

Output:



```
rohith@Rohith:~/Desktop/OSlab/Assignment-1$ gcc 5.c -o run
rohith@Rohith:~/Desktop/OSlab/Assignment-1$ ./run
Current ID: 993  Parent ID: 12
Current ID: 994  Parent ID: 993
No children to wait...
994 finished execution
```

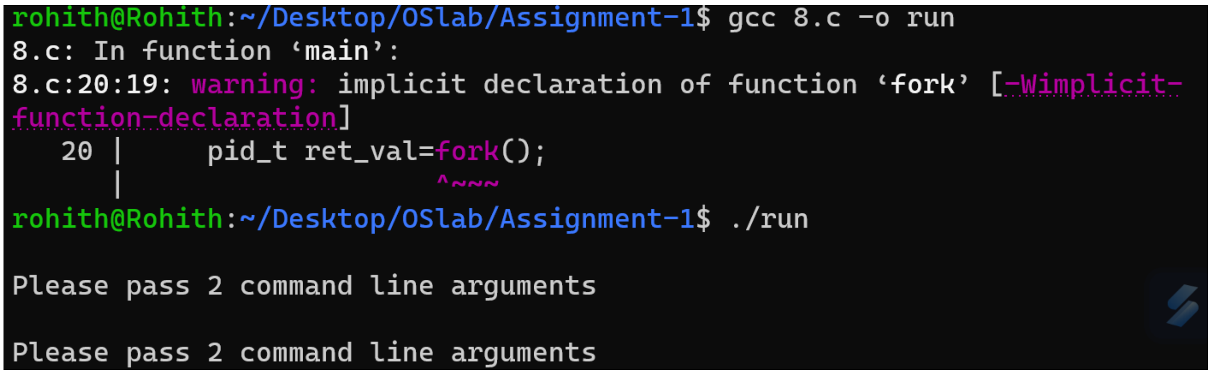
Explanation:

- Hence in the programs that use fork(), the fork() should wait for the child process to finish before the parent process gets terminated.
- No children to wait... is for the child process and 994 finished execution is for the parent process.
- wait() returns the process id you have waited for.

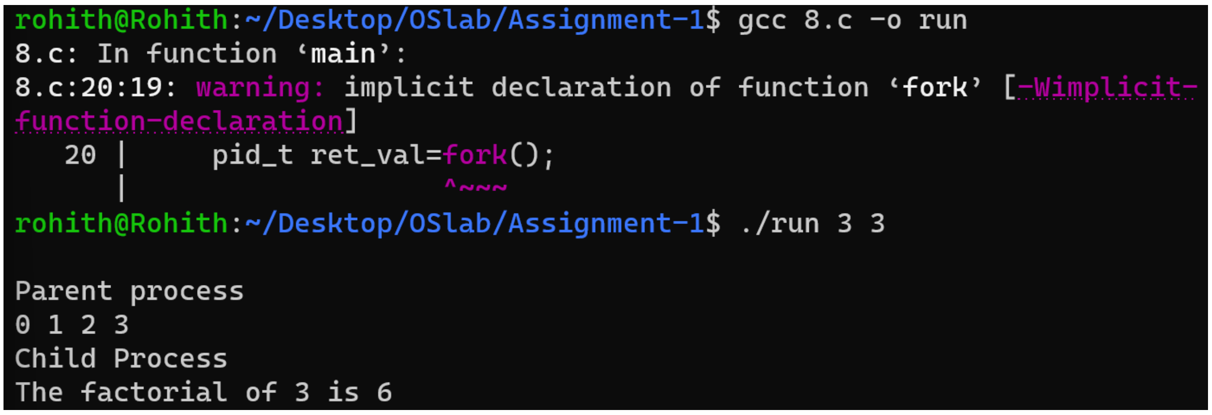
3. **Program code:**

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

long factorial(int num)
{
    if(num==0)
        return 1;
    return (factorial(num-1)*num);
}

void series(int num)
{
    for(int i=0;i<=num;i++)
        printf("%d ",i);
}

int main(int argc,char *argv[])
{
    pid_t ret_val=fork();
    if(argc==3)
    {
        if(ret_val<0)
        {
            //fork has failed
            printf("\nFork failure\n");
        }
        else if(ret_val==0)
        {
            printf("\nChild Process\n");
            printf("The factorial of %d is
%ld\n",atoi(argv[1]),factorial(atoi(argv[1])));
        }
        else
        {
            printf("\nParent process\n");
            series(atoi(argv[2]));
        }
    }
    else
    {
        printf("\nPlease pass 2 command line arguments\n");
    }
}
```

**Output:**



```
rohith@Rohith:~/Desktop/OSlab/Assignment-1$ gcc 8.c -o run
8.c: In function ‘main’:
8.c:20:19: warning: implicit declaration of function ‘fork’ [-Wimplicit-
function-declaration]
   20 |     pid_t ret_val=fork();
      |                   ^~~~
rohith@Rohith:~/Desktop/OSlab/Assignment-1$ ./run

Please pass 2 command line arguments

Please pass 2 command line arguments
```



```
rohith@Rohith:~/Desktop/OSlab/Assignment-1$ gcc 8.c -o run
8.c: In function ‘main’:
8.c:20:19: warning: implicit declaration of function ‘fork’ [-Wimplicit-
function-declaration]
   20 |     pid_t ret_val=fork();
      |                   ^~~~
rohith@Rohith:~/Desktop/OSlab/Assignment-1$ ./run 3 3

Parent process
0 1 2 3
Child Process
The factorial of 3 is 6
```