Assignment No. 8

a) Write a C program to create a child process and allow the parent to display 'parent' and the child to display ,child' on the screen.

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

int main() {
    pid_t pid = fork();

    if (pid == 0) {
        printf("Child process: child\n");
    } else if (pid > 0) {
        printf("Parent process: parent\n");
    } else {
        perror("fork");
        return 1;
    }

    return 0;
}
```

```
anubhav@DESKTOP-9VIA8NE:~/anubhav_oslab$ vim 8thAssigna.c
anubhav@DESKTOP-9VIA8NE:~/anubhav_oslab$ gcc -o 8thAssigna 8thAssigna.c
anubhav@DESKTOP-9VIA8NE:~/anubhav_oslab$ ./8thAssigna .
Parent process: parent
Child process: child
anubhav@DESKTOP-9VIA8NE:~/anubhav_oslab$
```

b) Write a C program to create a Zombie and orphan process.

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

int main() {
```

```
    pid_t child_pid = fork();

    if (child_pid == 0) {
        // Child process
        printf("Child process: sleeping...\n");
        sleep(5);
        printf("Child process: exiting.\n");
        exit(0);
    } else if (child_pid > 0) {
        // Parent process
        printf("Parent process: waiting for
child...\n");
        wait(NULL); // Wait for any child process to
exit
        printf("Parent process: exiting.\n");
    } else {
        perror("fork");
        return 1;
    }

    return 0;
}
```

```
anubhav@DESKTOP-9VIA8NE:~/anubhav_oslab$ vim 8thAssignb.c
anubhav@DESKTOP-9VIA8NE:~/anubhav_oslab$ gcc -o 8thAssignb 8thAssignb.c
anubhav@DESKTOP-9VIA8NE:~/anubhav_oslab$ ./8thAssignb
Parent process: waiting for child...
Child process: sleeping...
Child process: exiting.
Parent process: exiting.
anubhav@DESKTOP-9VIA8NE:~/anubhav_oslab$
```

c) Write a program that illustrates how to execute two commands concurrently.

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

int main() {
    pid_t pid = fork();

    if (pid == 0) {
        // Child process
        execlp("ls", "ls", "-l", NULL);
    } else if (pid > 0) {
```

```
        // Parent process
        printf("Parent process: Waiting for
child...\n");
        wait(NULL);
        printf("Parent process: Child process has
completed.\n");
    } else {
        perror("fork");
        return 1;
    }

    return 0;
}

anubhav@DESKTOP-9VIA8NE:~/anubhav_oslab$ rm
8thAssignc.c
anubhav@DESKTOP-9VIA8NE:~/anubhav_oslab$ vim
8thAssignc.c
anubhav@DESKTOP-9VIA8NE:~/anubhav_oslab$ gcc -o
8thAssignc 8thAssignc.c
anubhav@DESKTOP-9VIA8NE:~/anubhav_oslab$ ./8thAssignc
Parent process: Waiting for child...
total 252
-rwxr-xr-x 1 anubhav anubhav   216 Sep 20 10:37
3rdAssigna.sh
-rwxr-xr-x 1 anubhav anubhav   274 Sep 20 10:39
3rdAssignb.sh
-rw-r--r-- 1 anubhav anubhav   258 Aug 24 09:29
3rdAssignc.sh
-rw-r--r-- 1 anubhav anubhav   344 Aug 24 09:30
3rdAssignd.sh
-rw-r--r-- 1 anubhav anubhav  1024 Aug 24 09:32
3rdassignfile.txt
-rw-r--r-- 1 anubhav anubhav   635 Aug 24 18:48
4thassigna.sh
-rw-r--r-- 1 anubhav anubhav   330 Aug 24 18:50
4thassignb.sh
-rw-r--r-- 1 anubhav anubhav   496 Aug 24 18:53
4thassignc.sh
-rw-r--r-- 1 anubhav anubhav   619 Aug 24 19:16
4thassignd.sh
-rw-r--r-- 1 anubhav anubhav   233 Aug 24 19:21
5thassigna.sh
```

```
-rw-r--r-- 1 anubhav anubhav   332 Aug 24 19:32
5thassignb.sh
-rw-r--r-- 1 anubhav anubhav   201 Aug 24 19:37
5thassignc.sh
-rw-r--r-- 1 anubhav anubhav   750 Sep 21 11:13
7thAssigna.c
-rw-r--r-- 1 anubhav anubhav   442 Sep 21 11:17
7thAssignbi.c
-rw-r--r-- 1 anubhav anubhav   702 Sep 21 11:21
7thAssignbii.c
-rwxr-xr-x 1 anubhav anubhav 16184 Sep 21 11:26
-rwxr-xr-x 1 anubhav anubhav 16264 Sep 21 11:21
my_cp
-rw-r--r-- 1 anubhav anubhav   320 Sep 14 09:19
newemp.txt
-rw-r--r-- 1 anubhav anubhav   320 Sep 21 11:21
newfile.txt
-rw-r--r-- 1 anubhav anubhav    26 Sep 21 11:12
source.txt
-rw-r--r-- 1 anubhav anubhav   671 Sep 21 09:01
trace.txt
Parent process: Child process has completed.
```

d) Write a C program that illustrates suspending and resuming processes using signals.

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

void handler(int signum) {
    printf("Signal received: %d\n", signum);
}

int main() {
    signal(SIGUSR1, handler);

    pid_t child_pid = fork();

    if (child_pid == 0) {
        // Child process
        printf("Child process: sending signal to
parent...\n");
```

```
        kill(getppid(), SIGUSR1);
        sleep(2);
        printf("Child process: exiting.\n");
    } else if (child_pid > 0) {
        // Parent process
        printf("Parent process: waiting for
signal...\n");
        int status;
        wait(&status); // Wait for the child process
to finish
        printf("Parent process: signal received.\n");
    } else {
        perror("fork");
        return 1;
    }

    return 0;
}
```

```
anubhav@DESKTOP-9VIA8NE:~/anubhav_oslab$ rm 8thAssignd.c
anubhav@DESKTOP-9VIA8NE:~/anubhav_oslab$ vim 8thAssignd.c
anubhav@DESKTOP-9VIA8NE:~/anubhav_oslab$ gcc -o 8thAssignd 8thAssignd.c
anubhav@DESKTOP-9VIA8NE:~/anubhav_oslab$ ./8thAssignd
Parent process: waiting for signal...
Child process: sending signal to parent...
Signal received: 10
Child process: exiting.
Parent process: signal received.
anubhav@DESKTOP-9VIA8NE:~/anubhav_oslab$
```