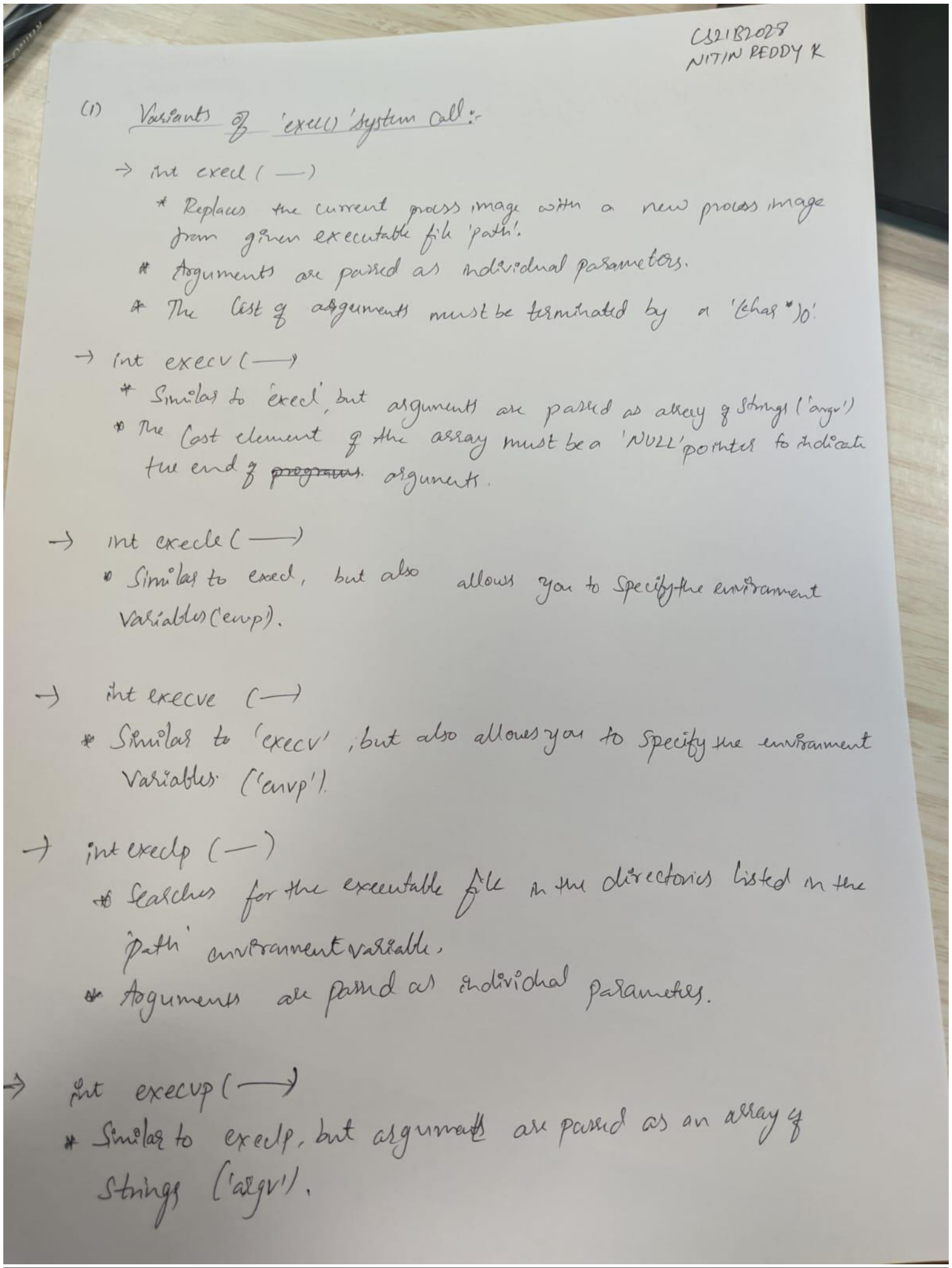
NITIN REDDY K

CS21B2028

Written work:



CS21B2028
NITIN REDDY K

(1) Variants of 'exec()' system call:-

→ int execl (—)

* Replaces the current process image with a new process image from given executable file 'path'.
* Arguments are passed as individual parameters.
* The list of arguments must be terminated by a '(char *)0'.

→ int execv (⟶)

* Similar to 'execl', but arguments are passed as array of strings ('argv')
* The last element of the array must be a 'NULL' pointer to indicate the end of ~~programs~~ arguments.

→ int execle (⟶)

* Similar to execl, but also allows you to specify the environment variables (envp).

→ int execve (⟶)

* Similar to 'execv', but also allows you to specify the environment variables ('envp').

→ int execlp (—)

* Searches for the executable file in the directories listed in the 'path' environment variable.
* Arguments are passed as individual parameters.

→ int execvp (⟶)

* Similar to execlp, but arguments are passed as an array of strings ('argv').

CS21B2028
NITIN REDDY K

Variants of wait() system call

(1) 'pid_t wait(int *status);

* Suspends execution of the calling process until one of its child process terminates.
* Returns the process ID of the terminated child process
* The exit status of the terminated child process is stored in 'status'.

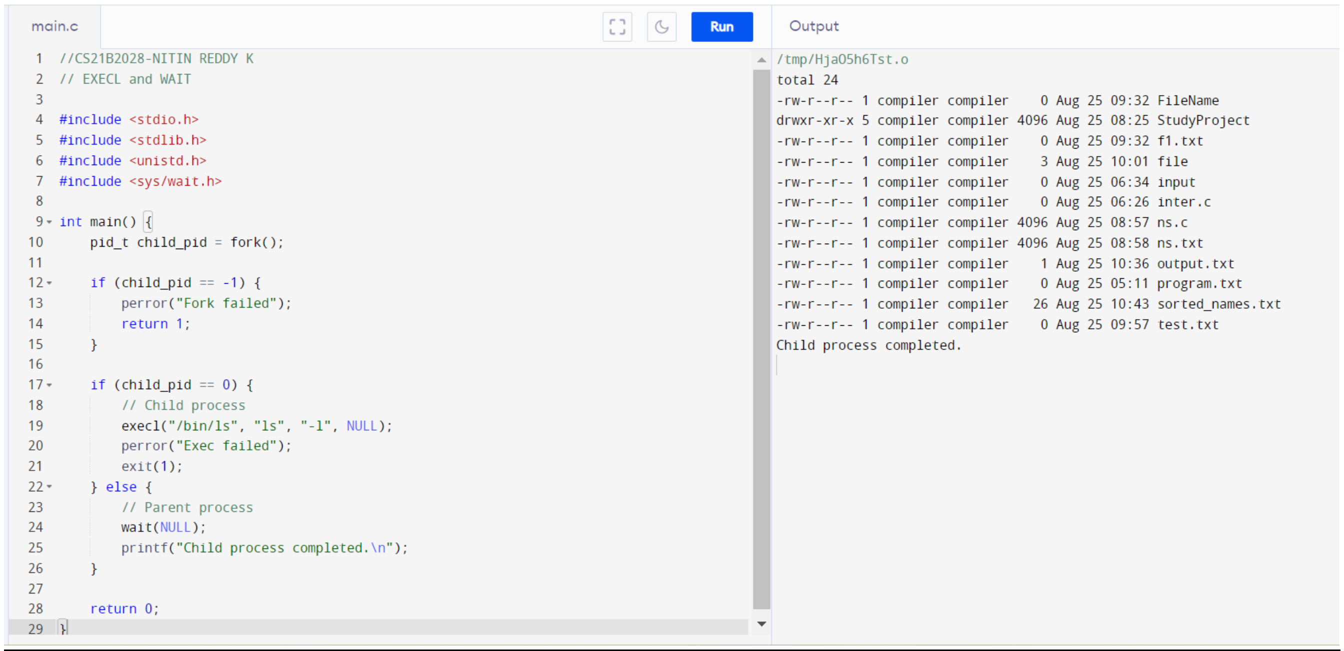
(2) pid_t waitpid(——);

* Suspends execution of the calling process until the child process specified by 'pid' terminates.
* Returns the process ID of the terminated child process.
* The exit status of the terminated child process is stored in 'status'.
* The options parameter can be used to customize the behaviour

Codes:

1)execl() and wait()

2)execv() and waitpid()



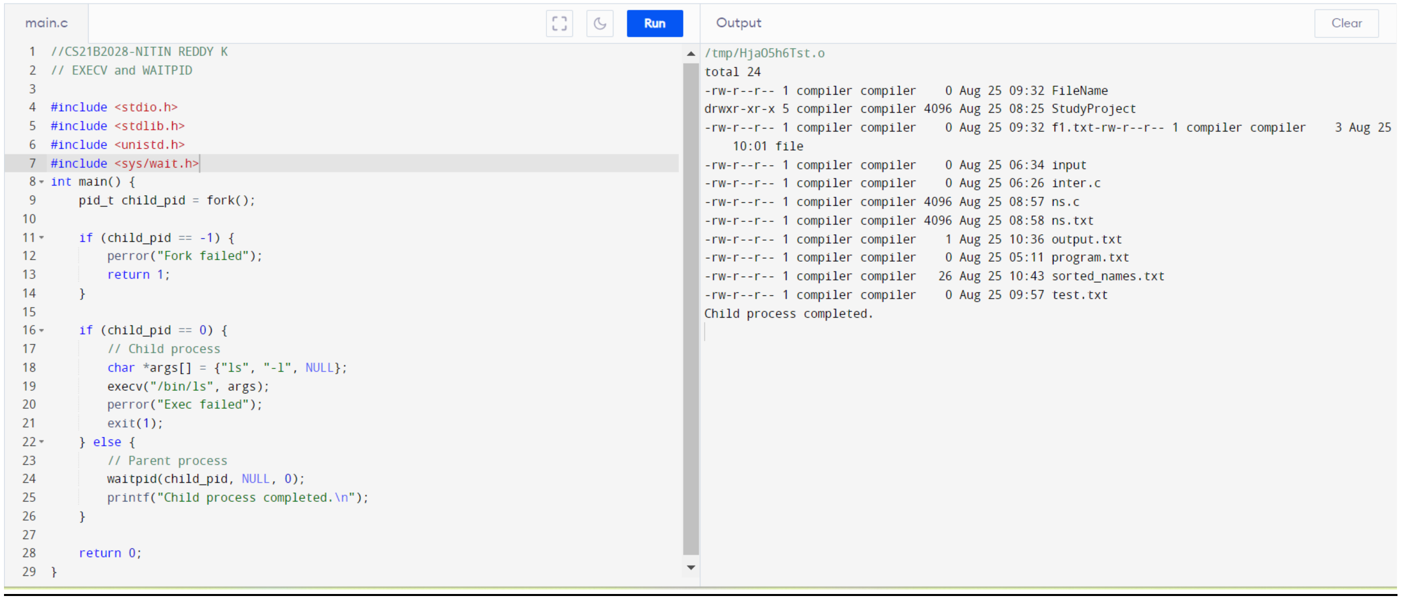
main.c

Run

Output

```c
//CS21B2028-NITIN REDDY K
// EXECL and WAIT

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

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

    if (child_pid == -1) {
        perror("Fork failed");
        return 1;
    }

    if (child_pid == 0) {
        // Child process
        execl("/bin/ls", "ls", "-l", NULL);
        perror("Exec failed");
        exit(1);
    } else {
        // Parent process
        wait(NULL);
        printf("Child process completed.\n");
    }

    return 0;
}
```

```
/tmp/HjaO5h6Tst.o
total 24
-rw-r--r-- 1 compiler compiler    0 Aug 25 09:32 FileName
drwxr-xr-x 5 compiler compiler 4096 Aug 25 08:25 StudyProject
-rw-r--r-- 1 compiler compiler    0 Aug 25 09:32 f1.txt
-rw-r--r-- 1 compiler compiler    3 Aug 25 10:01 file
-rw-r--r-- 1 compiler compiler    0 Aug 25 06:34 input
-rw-r--r-- 1 compiler compiler    0 Aug 25 06:26 inter.c
-rw-r--r-- 1 compiler compiler 4096 Aug 25 08:57 ns.c
-rw-r--r-- 1 compiler compiler 4096 Aug 25 08:58 ns.txt
-rw-r--r-- 1 compiler compiler    1 Aug 25 10:36 output.txt
-rw-r--r-- 1 compiler compiler    0 Aug 25 05:11 program.txt
-rw-r--r-- 1 compiler compiler   26 Aug 25 10:43 sorted_names.txt
-rw-r--r-- 1 compiler compiler    0 Aug 25 09:57 test.txt
Child process completed.
```



main.c

Run

Output

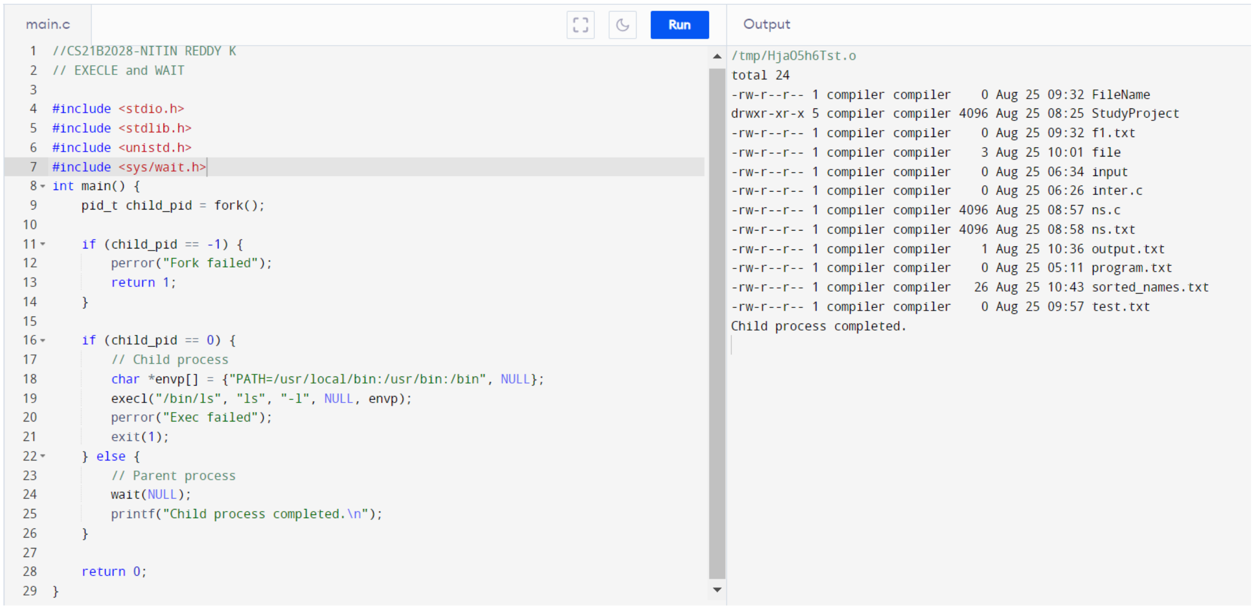
Clear

```c
//CS21B2028-NITIN REDDY K
// EXECV and WAITPID

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>
int main() {
    pid_t child_pid = fork();

    if (child_pid == -1) {
        perror("Fork failed");
        return 1;
    }

    if (child_pid == 0) {
        // Child process
        char *args[] = {"ls", "-l", NULL};
        execv("/bin/ls", args);
        perror("Exec failed");
        exit(1);
    } else {
        // Parent process
        waitpid(child_pid, NULL, 0);
        printf("Child process completed.\n");
    }

    return 0;
}
```

```
/tmp/HjaO5h6Tst.o
total 24
-rw-r--r-- 1 compiler compiler    0 Aug 25 09:32 FileName
drwxr-xr-x 5 compiler compiler 4096 Aug 25 08:25 StudyProject
-rw-r--r-- 1 compiler compiler    0 Aug 25 09:32 f1.txt-rw-r--r-- 1 compiler compiler    3 Aug 25
    10:01 file
-rw-r--r-- 1 compiler compiler    0 Aug 25 06:34 input
-rw-r--r-- 1 compiler compiler    0 Aug 25 06:26 inter.c
-rw-r--r-- 1 compiler compiler 4096 Aug 25 08:57 ns.c
-rw-r--r-- 1 compiler compiler 4096 Aug 25 08:58 ns.txt
-rw-r--r-- 1 compiler compiler    1 Aug 25 10:36 output.txt
-rw-r--r-- 1 compiler compiler    0 Aug 25 05:11 program.txt
-rw-r--r-- 1 compiler compiler   26 Aug 25 10:43 sorted_names.txt
-rw-r--r-- 1 compiler compiler    0 Aug 25 09:57 test.txt
Child process completed.
```

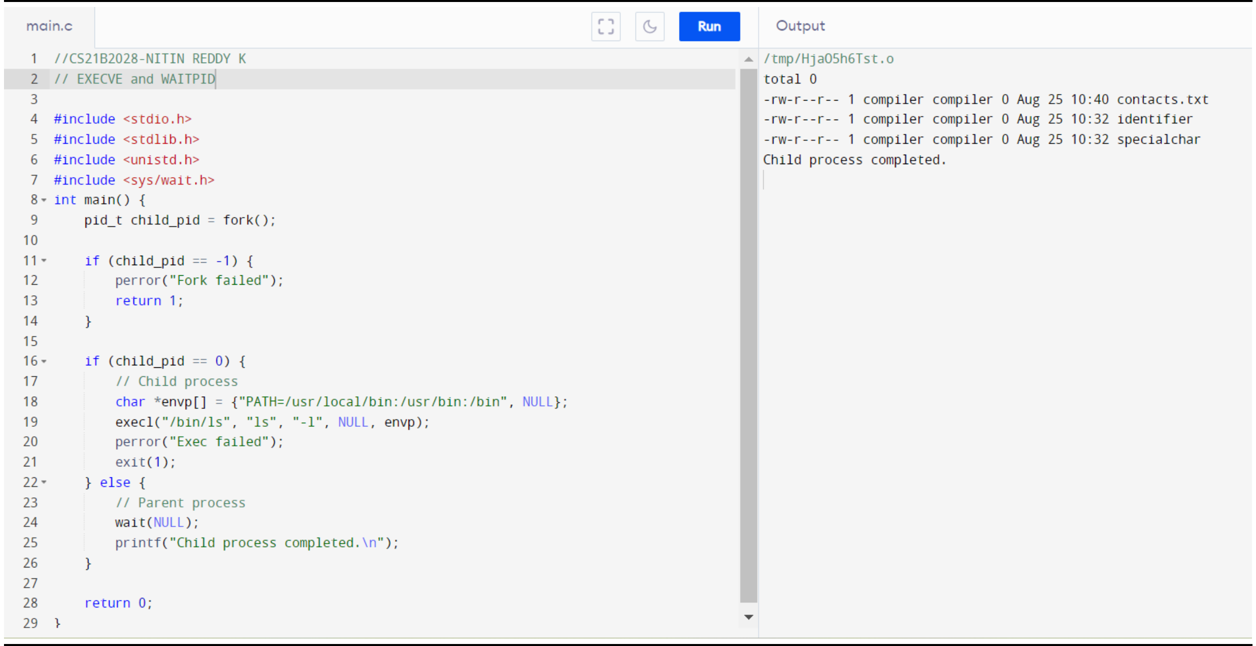
3)execle() and wait()

4)execve() and waitpid()



```c
//CS21B2028-NITIN REDDY K
// EXECLE and WAIT

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>
int main() {
    pid_t child_pid = fork();

    if (child_pid == -1) {
        perror("Fork failed");
        return 1;
    }

    if (child_pid == 0) {
        // Child process
        char *envp[] = {"PATH=/usr/local/bin:/usr/bin:/bin", NULL};
        execl("/bin/ls", "ls", "-l", NULL, envp);
        perror("Exec failed");
        exit(1);
    } else {
        // Parent process
        wait(NULL);
        printf("Child process completed.\n");
    }

    return 0;
}
```

```
/tmp/Hja05h6Tst.o
total 24
-rw-r--r-- 1 compiler compiler    0 Aug 25 09:32 FileName
drwxr-xr-x 5 compiler compiler 4096 Aug 25 08:25 StudyProject
-rw-r--r-- 1 compiler compiler    0 Aug 25 09:32 f1.txt
-rw-r--r-- 1 compiler compiler    3 Aug 25 10:01 file
-rw-r--r-- 1 compiler compiler    0 Aug 25 06:34 input
-rw-r--r-- 1 compiler compiler    0 Aug 25 06:26 inter.c
-rw-r--r-- 1 compiler compiler 4096 Aug 25 08:57 ns.c
-rw-r--r-- 1 compiler compiler 4096 Aug 25 08:58 ns.txt
-rw-r--r-- 1 compiler compiler    1 Aug 25 10:36 output.txt
-rw-r--r-- 1 compiler compiler    0 Aug 25 05:11 program.txt
-rw-r--r-- 1 compiler compiler   26 Aug 25 10:43 sorted_names.txt
-rw-r--r-- 1 compiler compiler    0 Aug 25 09:57 test.txt
Child process completed.
```

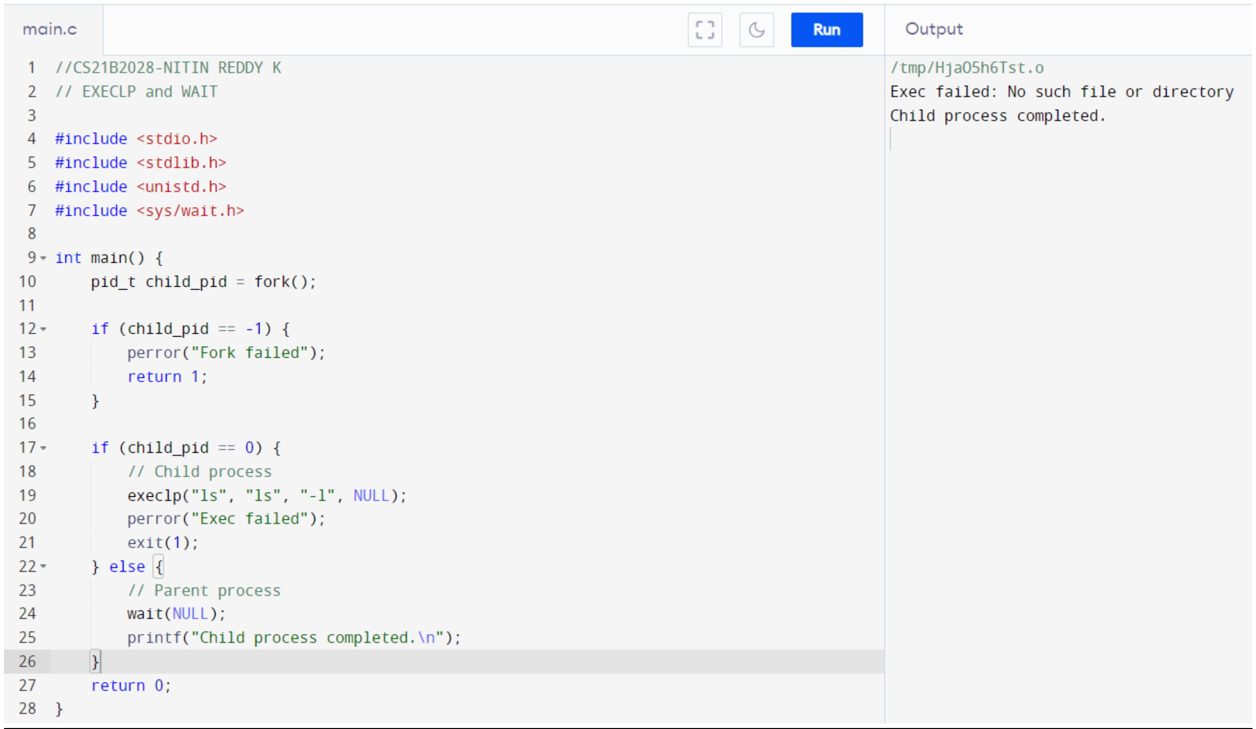


```c
//CS21B2028-NITIN REDDY K
// EXECVE and WAITPID

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>
int main() {
    pid_t child_pid = fork();

    if (child_pid == -1) {
        perror("Fork failed");
        return 1;
    }

    if (child_pid == 0) {
        // Child process
        char *envp[] = {"PATH=/usr/local/bin:/usr/bin:/bin", NULL};
        execl("/bin/ls", "ls", "-l", NULL, envp);
        perror("Exec failed");
        exit(1);
    } else {
        // Parent process
        wait(NULL);
        printf("Child process completed.\n");
    }

    return 0;
}
```

```
/tmp/Hja05h6Tst.o
total 0
-rw-r--r-- 1 compiler compiler 0 Aug 25 10:40 contacts.txt
-rw-r--r-- 1 compiler compiler 0 Aug 25 10:32 identifier
-rw-r--r-- 1 compiler compiler 0 Aug 25 10:32 specialchar
Child process completed.
```

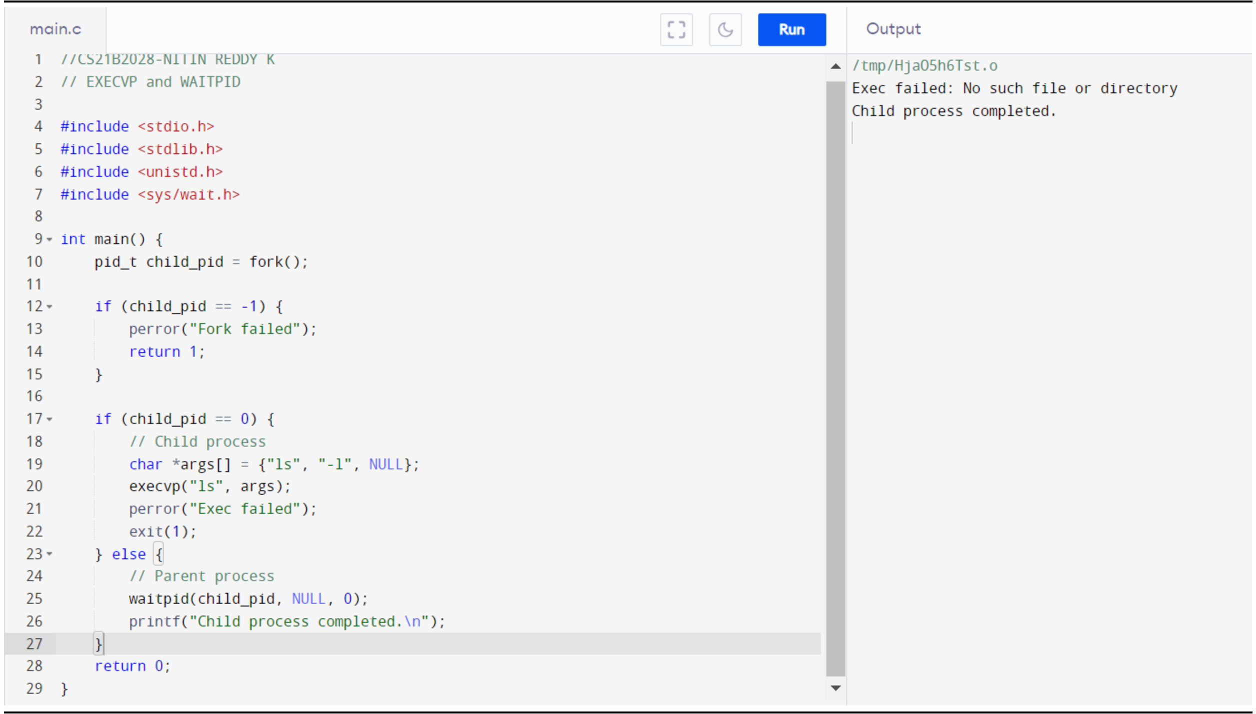
5)execlp() and wait()

6)execvp() and waitpid()



```c
//CS21B2028-NITIN REDDY K
// EXECLP and WAIT

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

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

    if (child_pid == -1) {
        perror("Fork failed");
        return 1;
    }

    if (child_pid == 0) {
        // Child process
        execlp("ls", "ls", "-l", NULL);
        perror("Exec failed");
        exit(1);
    } else {
        // Parent process
        wait(NULL);
        printf("Child process completed.\n");
    }
    return 0;
}
```

```
/tmp/Hja05h6Tst.o
Exec failed: No such file or directory
Child process completed.
```

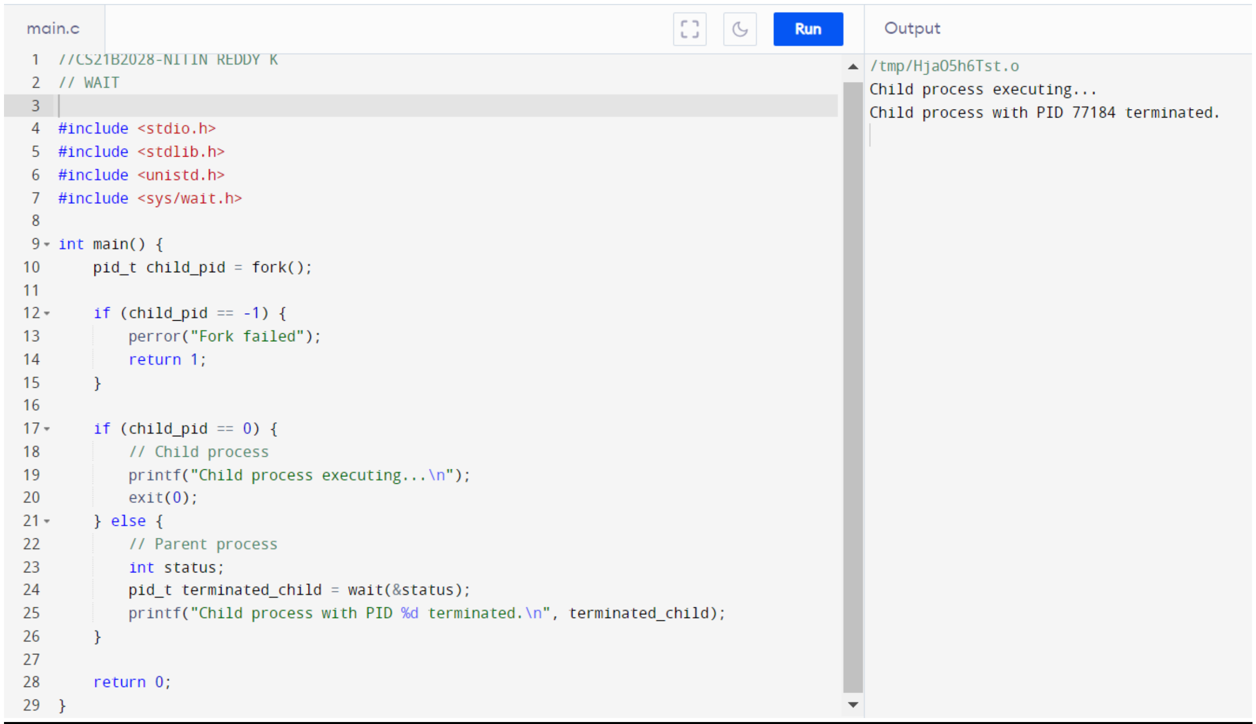


```c
//CS21B2028-NITIN REDDY K
// EXECVP and WAITPID

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

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

    if (child_pid == -1) {
        perror("Fork failed");
        return 1;
    }

    if (child_pid == 0) {
        // Child process
        char *args[] = {"ls", "-l", NULL};
        execvp("ls", args);
        perror("Exec failed");
        exit(1);
    } else {
        // Parent process
        waitpid(child_pid, NULL, 0);
        printf("Child process completed.\n");
    }
    return 0;
}
```

```
/tmp/Hja05h6Tst.o
Exec failed: No such file or directory
Child process completed.
```

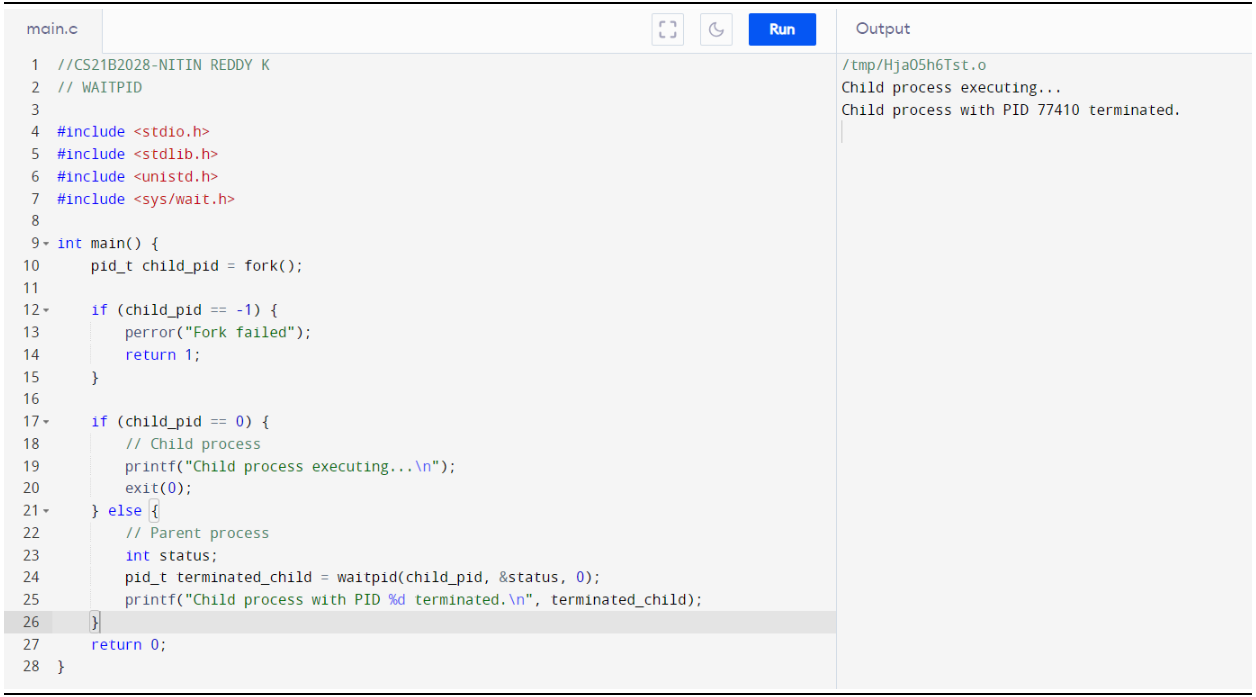
7)wait()

8)waitpid()



```c
//CS21B2028-NITIN REDDY K
// WAIT

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

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

    if (child_pid == -1) {
        perror("Fork failed");
        return 1;
    }

    if (child_pid == 0) {
        // Child process
        printf("Child process executing...\n");
        exit(0);
    } else {
        // Parent process
        int status;
        pid_t terminated_child = wait(&status);
        printf("Child process with PID %d terminated.\n", terminated_child);
    }

    return 0;
}
```

```
/tmp/Hja05h6Tst.o
Child process executing...
Child process with PID 77184 terminated.
```



```c
//CS21B2028-NITIN REDDY K
// WAITPID

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

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

    if (child_pid == -1) {
        perror("Fork failed");
        return 1;
    }

    if (child_pid == 0) {
        // Child process
        printf("Child process executing...\n");
        exit(0);
    } else {
        // Parent process
        int status;
        pid_t terminated_child = waitpid(child_pid, &status, 0);
        printf("Child process with PID %d terminated.\n", terminated_child);
    }
    return 0;
}
```

```
/tmp/Hja05h6Tst.o
Child process executing...
Child process with PID 77410 terminated.
```