# Task 6:

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

int main()

{

pid\_t c1, c2;

c1 = fork();

if (c1 == 0)

{

printf("I am C1 with PID: %d\n", getpid());

sleep(5);

printf("C1 is exiting\n");

exit(0);

}

else if (c1 > 0)

{

c2 = fork();

if (c2 == 0)

{

printf("I am C2 with PID: %d\n", getpid());

sleep(10);

printf("C2 is exiting\n");

exit(0);

}

else if (c2 > 0)

{

sleep(2);

printf("Parent process with PID: %d is exiting, leaving C2 an orphan.\n", getpid());

exit(0);

}

else

{

perror("Error creating C2");

exit(1);

}

}

else

{

perror("Error creating C1");

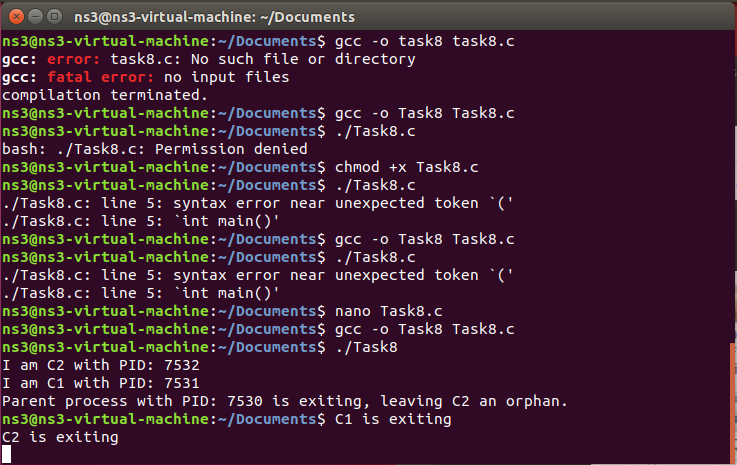
exit(1);

}

return 0;

}

# OUTPUT:



# TASK 7:

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/wait.h>

int main()

{

pid\_t c1, c2;

c1 = fork();

if (c1 == 0)

{

printf("I am C1 with PID: %d\n", getpid());

printf("C1 is exiting\n");

exit(0);

}

else if (c1 > 0)

{

c2 = fork();

if (c2 == 0)

{

printf("I am C2 with PID: %d\n", getpid());

printf("C2 is an orphan process\n");

sleep(10);

printf("C2 is exiting\n");

exit(0);

}

else if (c2 > 0)

{

sleep(2);

printf("Parent process with PID: %d is exiting, leaving C1 as a zombie.\n", getpid());

exit(0);

}

else

{

perror("Error creating C2");

exit(1);

}

}

else

{

perror("Error creating C1");

exit(1);

}

if (c1 > 0)

{

int status;

pid\_t pid = waitpid(c1, &status, 0);

if (pid == -1)

{

perror("Error waiting for C1");

}

else

{

printf("C1 with PID %d exited with status %d and became a zombie\n", pid, status);

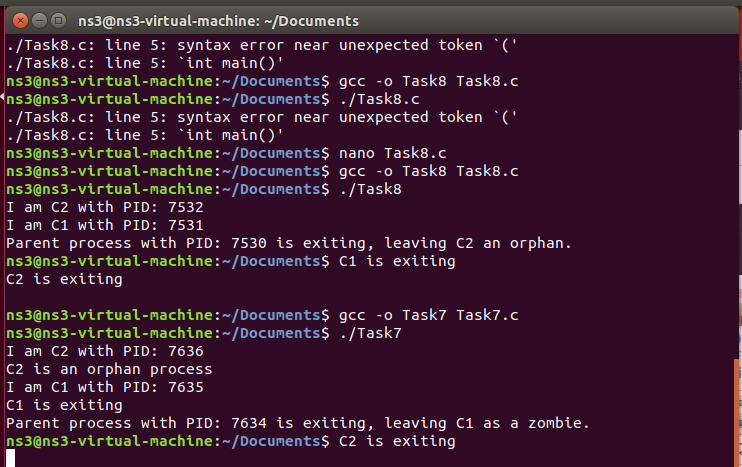
}

}

return 0;

}

# OUTPUT:



# TASK8:

#include <stdio.h>

#include <unistd.h>

#include <sys/wait.h>

int fibonacci(int n)

{

if (n <= 1)

{

return n;

}

else

{

return fibonacci(n - 1) + fibonacci(n - 2);

}

}

int factorial(int n)

{

if (n == 0)

{

return 1;

}

else

{

return n \* factorial(n - 1);

}

}

int main()

{

pid\_t pid;

int n, status;

printf("Enter a number to calculate its factorial: ");

scanf("%d", &n);

pid = fork();

if (pid == 0)

{

int result = factorial(n);

printf("Factorial of %d is %d\n", n, result);

} else if (pid > 0)

{

int steps;

printf("Enter the number of steps for the Fibonacci series: ");

scanf("%d", &steps);

int i, fib;

for (i = 0; i < steps; i++)

{

fib = fibonacci(i);

printf("%d ", fib);

}

printf("\n");

wait(&status);

} else

{

perror("Error creating child process");

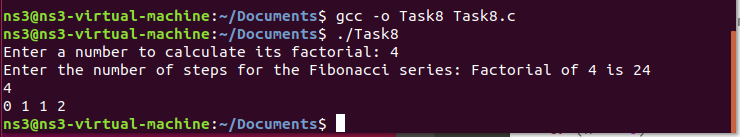
return 1;

}

return 0;

}

# OUTPUT:



# TASK9: #include <stdio.h>

#include <unistd.h>

#include <sys/wait.h>

int main() {

int choice;

do {

printf("Menu\n");

printf("1. EXEC VP\n");

printf("2. EXEC L\n");

printf("3. EXEC V??\n");

printf("4. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

execl("/bin/vp", "vp", NULL);

perror("exec failed");

break;

case 2:

execl("/bin/l", "l", NULL);

perror("exec failed");

break;

case 3:

execl("/bin/v??", "v??", NULL);

perror("exec failed");

break;

case 4:

printf("Exiting...\n");

break;

default:

printf("Invalid choice\n");

}

} while (choice != 4);

return 0;

}

# OUTPUT:

