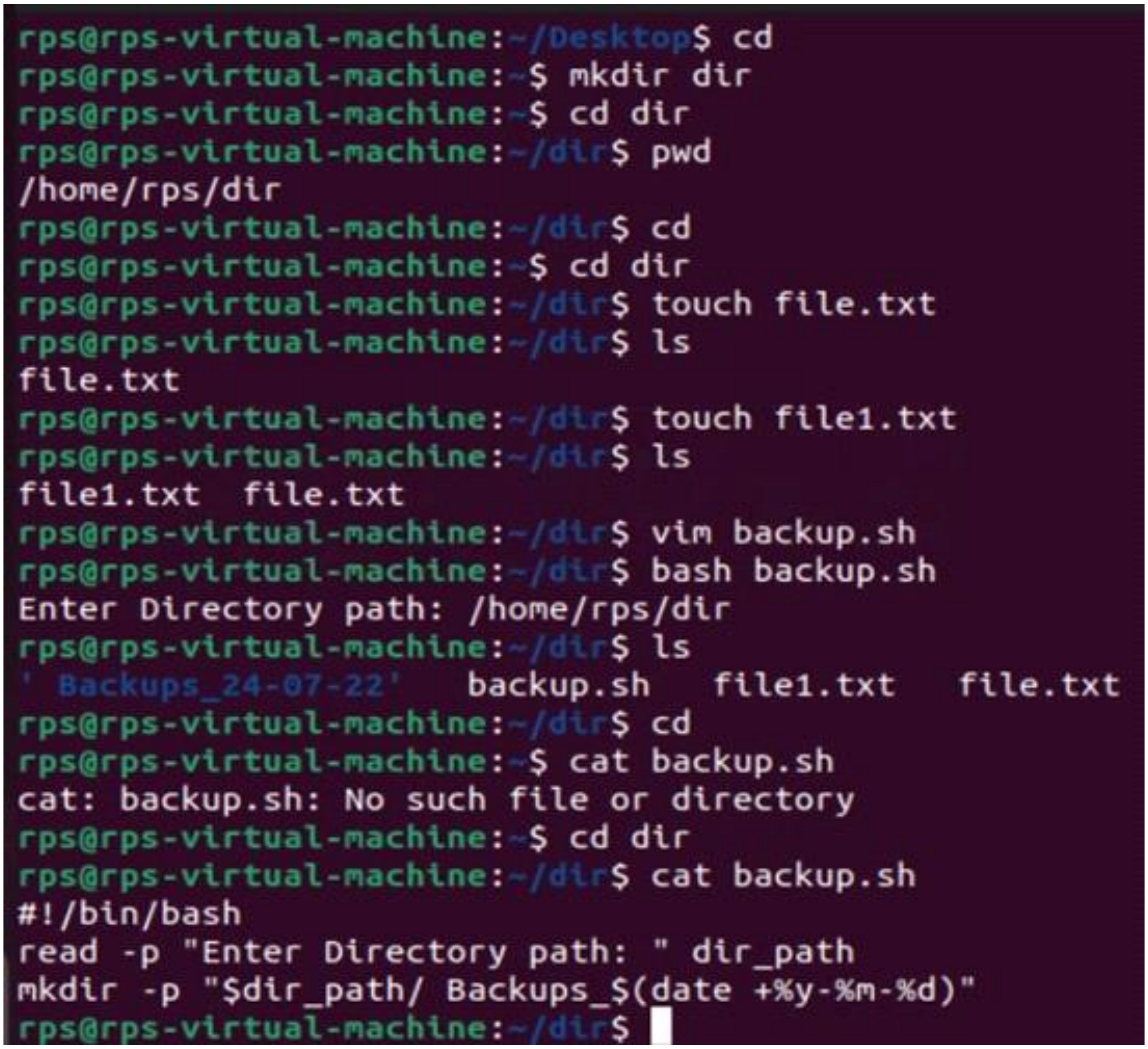
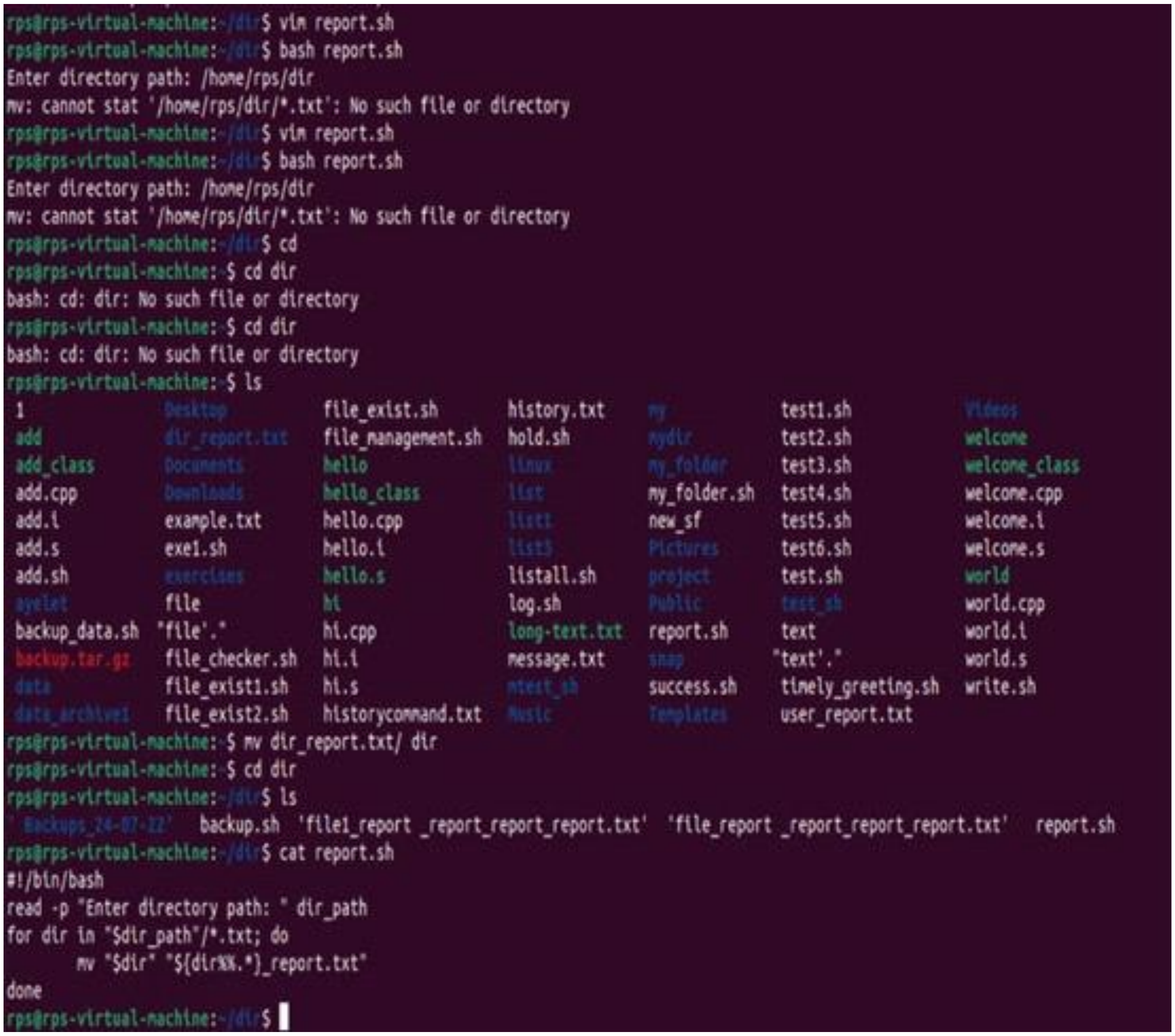
**1. File Management:**

**Write a script that takes a directory path as input and creates a new directory within it named "Backups_$(date +%Y-%m-%d)".**

**Create a script that renames all files in a directory with the extension ".txt" to have a prefix of "report_".**



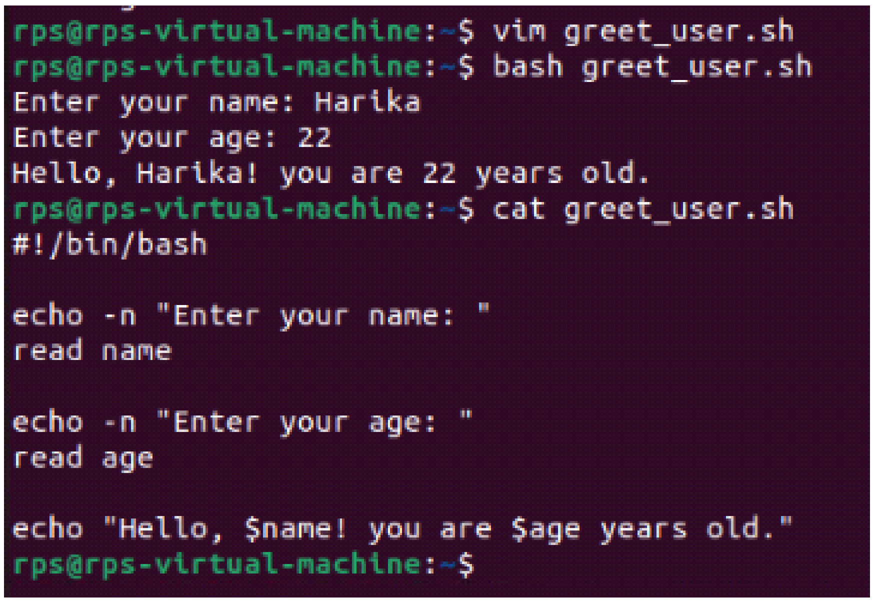
```
rps@rps-virtual-machine:~/Desktop$ cd
rps@rps-virtual-machine:~$ mkdir dir
rps@rps-virtual-machine:~$ cd dir
rps@rps-virtual-machine:~/dir$ pwd
/home/rps/dir
rps@rps-virtual-machine:~/dir$ cd
rps@rps-virtual-machine:~$ cd dir
rps@rps-virtual-machine:~/dir$ touch file.txt
rps@rps-virtual-machine:~/dir$ ls
file.txt
rps@rps-virtual-machine:~/dir$ touch file1.txt
rps@rps-virtual-machine:~/dir$ ls
file1.txt  file.txt
rps@rps-virtual-machine:~/dir$ vim backup.sh
rps@rps-virtual-machine:~/dir$ bash backup.sh
Enter Directory path: /home/rps/dir
rps@rps-virtual-machine:~/dir$ ls
' Backups_24-07-22'   backup.sh   file1.txt   file.txt
rps@rps-virtual-machine:~/dir$ cd
rps@rps-virtual-machine:~$ cat backup.sh
cat: backup.sh: No such file or directory
rps@rps-virtual-machine:~$ cd dir
rps@rps-virtual-machine:~/dir$ cat backup.sh
#!/bin/bash
read -p "Enter Directory path: " dir_path
mkdir -p "$dir_path/ Backups_$(date +%y-%m-%d)"
rps@rps-virtual-machine:~/dir$ 
```



```
rps@rps-virtual-machine:~/dir$ vim report.sh
rps@rps-virtual-machine:~/dir$ bash report.sh
Enter directory path: /home/rps/dir
mv: cannot stat '/home/rps/dir/*.txt': No such file or directory
rps@rps-virtual-machine:~/dir$ vim report.sh
rps@rps-virtual-machine:~/dir$ bash report.sh
Enter directory path: /home/rps/dir
mv: cannot stat '/home/rps/dir/*.txt': No such file or directory
rps@rps-virtual-machine:~/dir$ cd
rps@rps-virtual-machine:~$ cd dir
bash: cd: dir: No such file or directory
rps@rps-virtual-machine:~$ cd dir
bash: cd: dir: No such file or directory
rps@rps-virtual-machine:~$ ls
 1               Desktop           file_exist.sh        history.txt     my              test1.sh             Videos
 add             dir_report.txt    file_management.sh   hold.sh         mydir           test2.sh             welcome
 add_class       Documents         hello                linux           my_folder       test3.sh             welcome_class
 add.cpp         Downloads         hello_class          list            my_folder.sh    test4.sh             welcome.cpp
 add.i           example.txt       hello.cpp            list1           new_sf          test5.sh             welcome.i
 add.s           exe1.sh           hello.i              list3           Pictures        test6.sh             welcome.s
 add.sh          exercises         hello.s              listall.sh      project         test.sh              world
 ayelet          file              hi                   log.sh          Public          test_sh              world.cpp
 backup_data.sh "file'."           hi.cpp               long-text.txt   report.sh       text                 world.i
 backup.tar.gz   file_checker.sh   hi.i                 message.txt     snap           "text'."              world.s
 data            file_exist1.sh    hi.s                 ntest_sh        success.sh      timely_greeting.sh   write.sh
 data_archive1   file_exist2.sh    historycommand.txt   Music           Templates       user_report.txt
rps@rps-virtual-machine:~$ mv dir_report.txt/ dir
rps@rps-virtual-machine:~$ cd dir
rps@rps-virtual-machine:~/dir$ ls
' Backups_24-07-22'   backup.sh  'file1_report _report_report_report.txt'  'file_report _report_report_report.txt'   report.sh
rps@rps-virtual-machine:~/dir$ cat report.sh
#!/bin/bash
read -p "Enter directory path: " dir_path
for dir in "$dir_path"/*.txt; do
        mv "$dir" "${dir%%.*}_report.txt"
done
rps@rps-virtual-machine:~/dir$ 
```

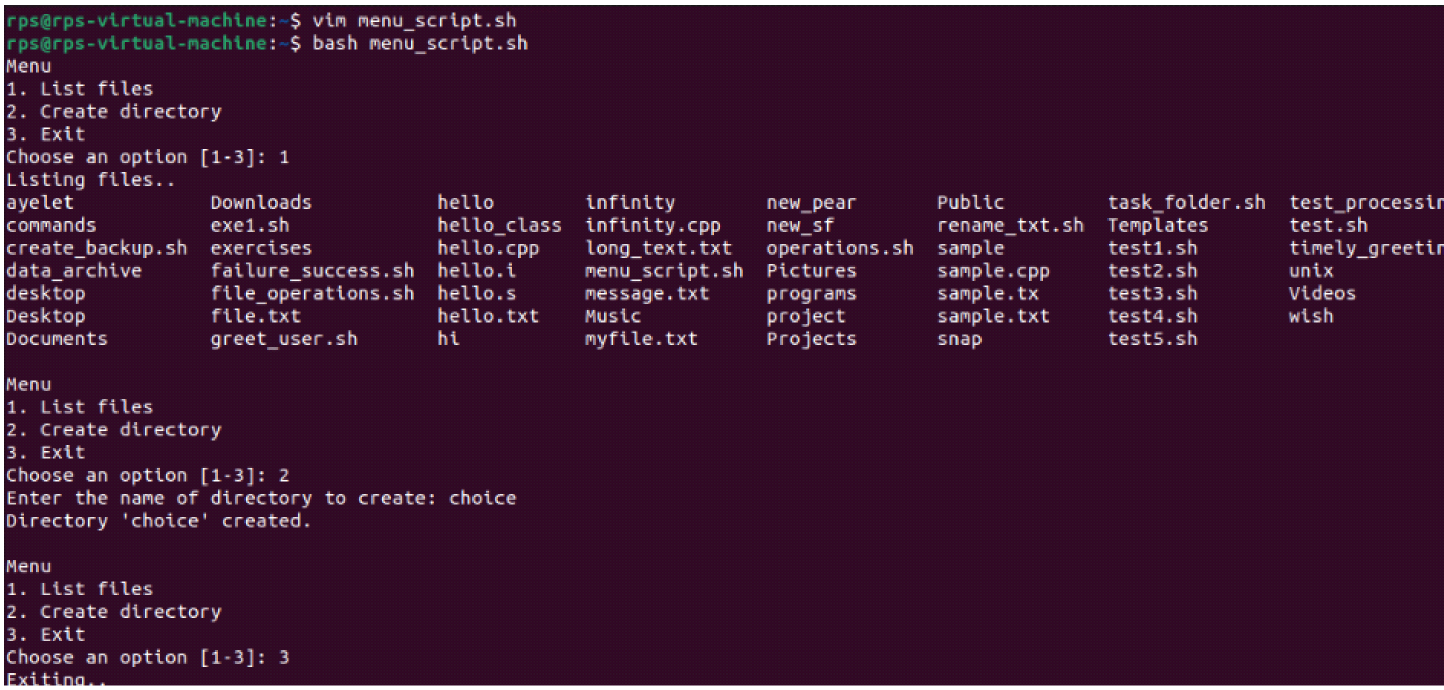
**2. User Interaction:**

**Write a script that prompts the user for their name and age, then greets them with a personalized message**.

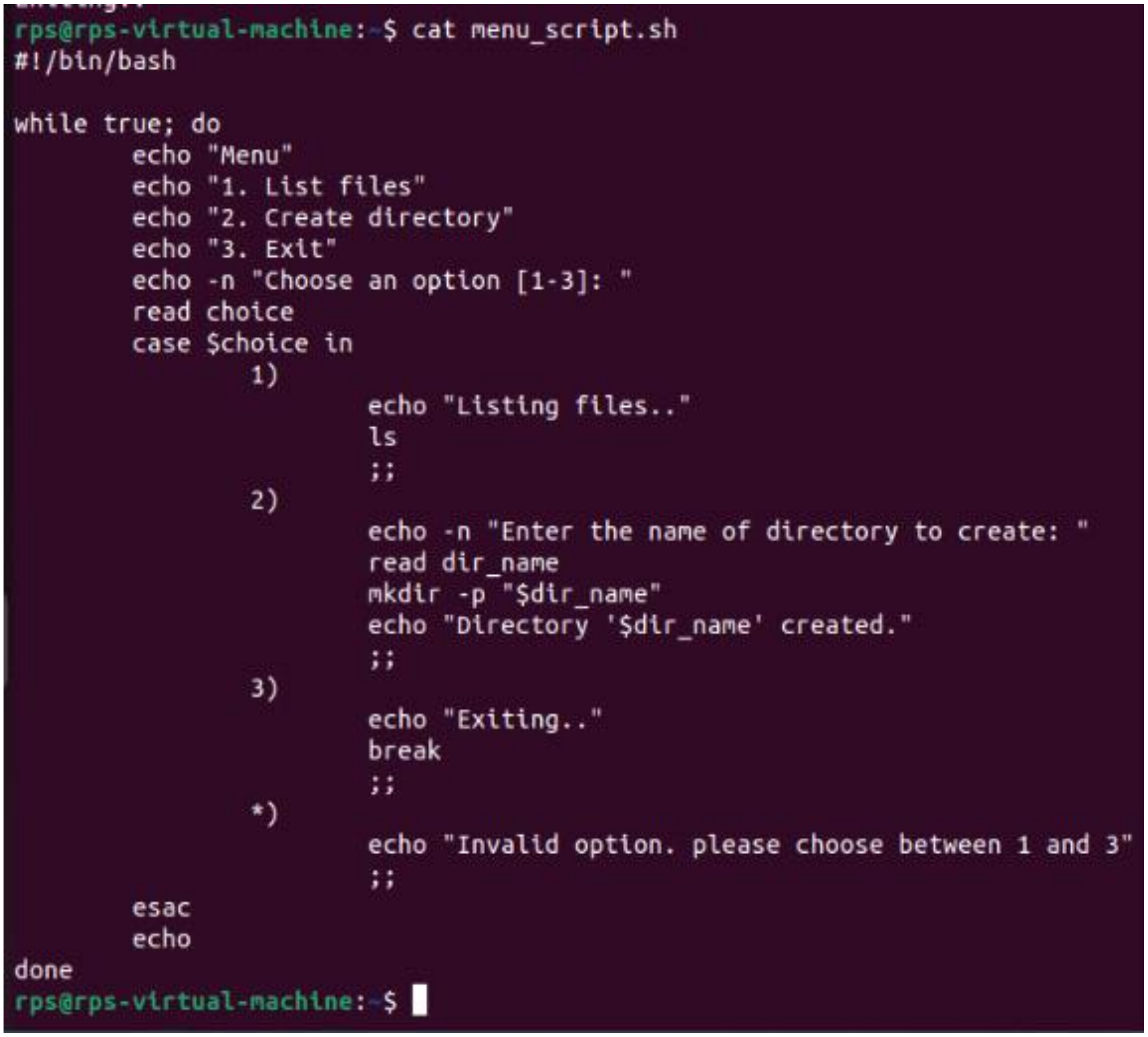


```
rps@rps-virtual-machine:~$ vim greet_user.sh
rps@rps-virtual-machine:~$ bash greet_user.sh
Enter your name: Harika
Enter your age: 22
Hello, Harika! you are 22 years old.
rps@rps-virtual-machine:~$ cat greet_user.sh
#!/bin/bash

echo -n "Enter your name: "
read name

echo -n "Enter your age: "
read age

echo "Hello, $name! you are $age years old."
rps@rps-virtual-machine:~$
```

**Design a script that displays a menu with options like "List files," "Create directory," and "Exit." Allow the user to choose an option and perform the corresponding action.**



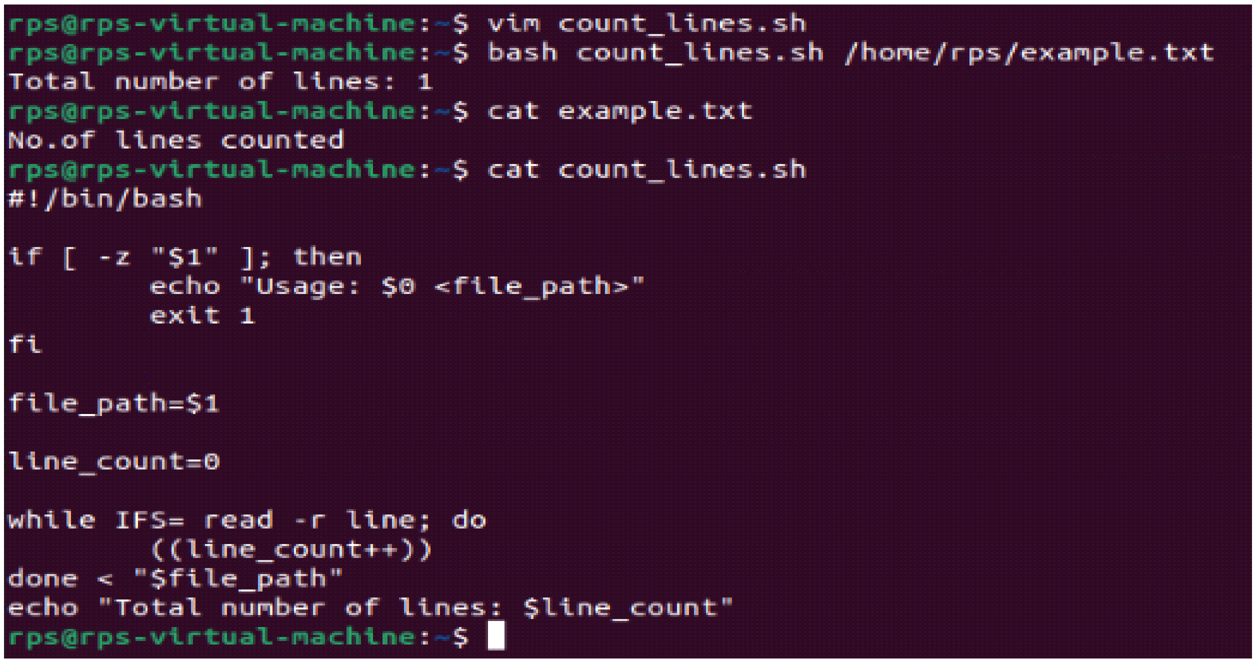
```
rps@rps-virtual-machine:~$ vim menu_script.sh
rps@rps-virtual-machine:~$ bash menu_script.sh
Menu
1. List files
2. Create directory
3. Exit
Choose an option [1-3]: 1
Listing files..
ayelet            Downloads           hello        infinity        new_pear       Public         task_folder.sh  test_processin
commands          exe1.sh             hello_class  infinity.cpp    new_sf         rename_txt.sh  Templates       test.sh
create_backup.sh  exercises           hello.cpp    long_text.txt   operations.sh  sample         test1.sh        timely_greetin
data_archive      failure_success.sh  hello.i      menu_script.sh  Pictures       sample.cpp     test2.sh        unix
desktop           file_operations.sh  hello.s      message.txt     programs       sample.tx      test3.sh        Videos
Desktop           file.txt            hello.txt    Music           project        sample.txt     test4.sh        wish
Documents         greet_user.sh       hi           myfile.txt      Projects       snap           test5.sh

Menu
1. List files
2. Create directory
3. Exit
Choose an option [1-3]: 2
Enter the name of directory to create: choice
Directory 'choice' created.

Menu
1. List files
2. Create directory
3. Exit
Choose an option [1-3]: 3
Exiting..
```



```
rps@rps-virtual-machine:~$ cat menu_script.sh
#!/bin/bash

while true; do
        echo "Menu"
        echo "1. List files"
        echo "2. Create directory"
        echo "3. Exit"
        echo -n "Choose an option [1-3]: "
        read choice
        case $choice in
                1)
                        echo "Listing files.."
                        ls
                        ;;
                2)
                        echo -n "Enter the name of directory to create: "
                        read dir_name
                        mkdir -p "$dir_name"
                        echo "Directory '$dir_name' created."
                        ;;
                3)
                        echo "Exiting.."
                        break
                        ;;
                *)
                        echo "Invalid option. please choose between 1 and 3"
                        ;;
        esac
        echo
done
rps@rps-virtual-machine:~$ 
```

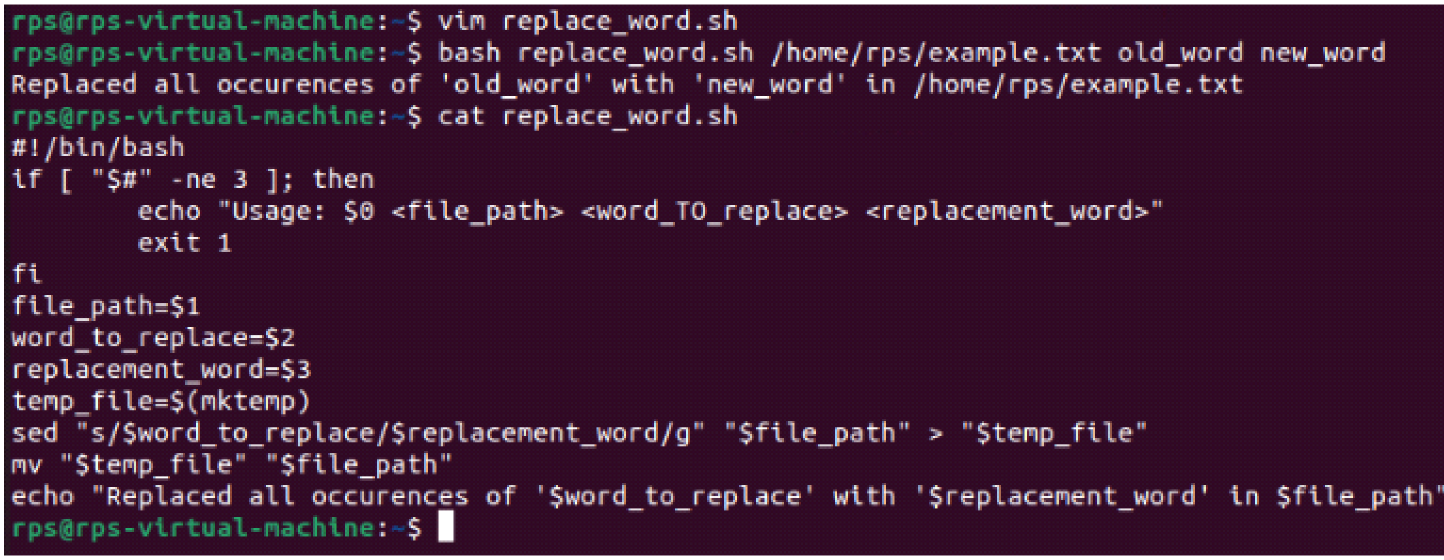
**3. Text Processing:**

**Write a script that reads the contents of a file line by line, counts the number of lines, and prints the total.**



```
rps@rps-virtual-machine:~$ vim count_lines.sh
rps@rps-virtual-machine:~$ bash count_lines.sh /home/rps/example.txt
Total number of lines: 1
rps@rps-virtual-machine:~$ cat example.txt
No.of lines counted
rps@rps-virtual-machine:~$ cat count_lines.sh
#!/bin/bash

if [ -z "$1" ]; then
        echo "Usage: $0 <file_path>"
        exit 1
fi

file_path=$1

line_count=0

while IFS= read -r line; do
        ((line_count++))
done < "$file_path"
echo "Total number of lines: $line_count"
rps@rps-virtual-machine:~$
```

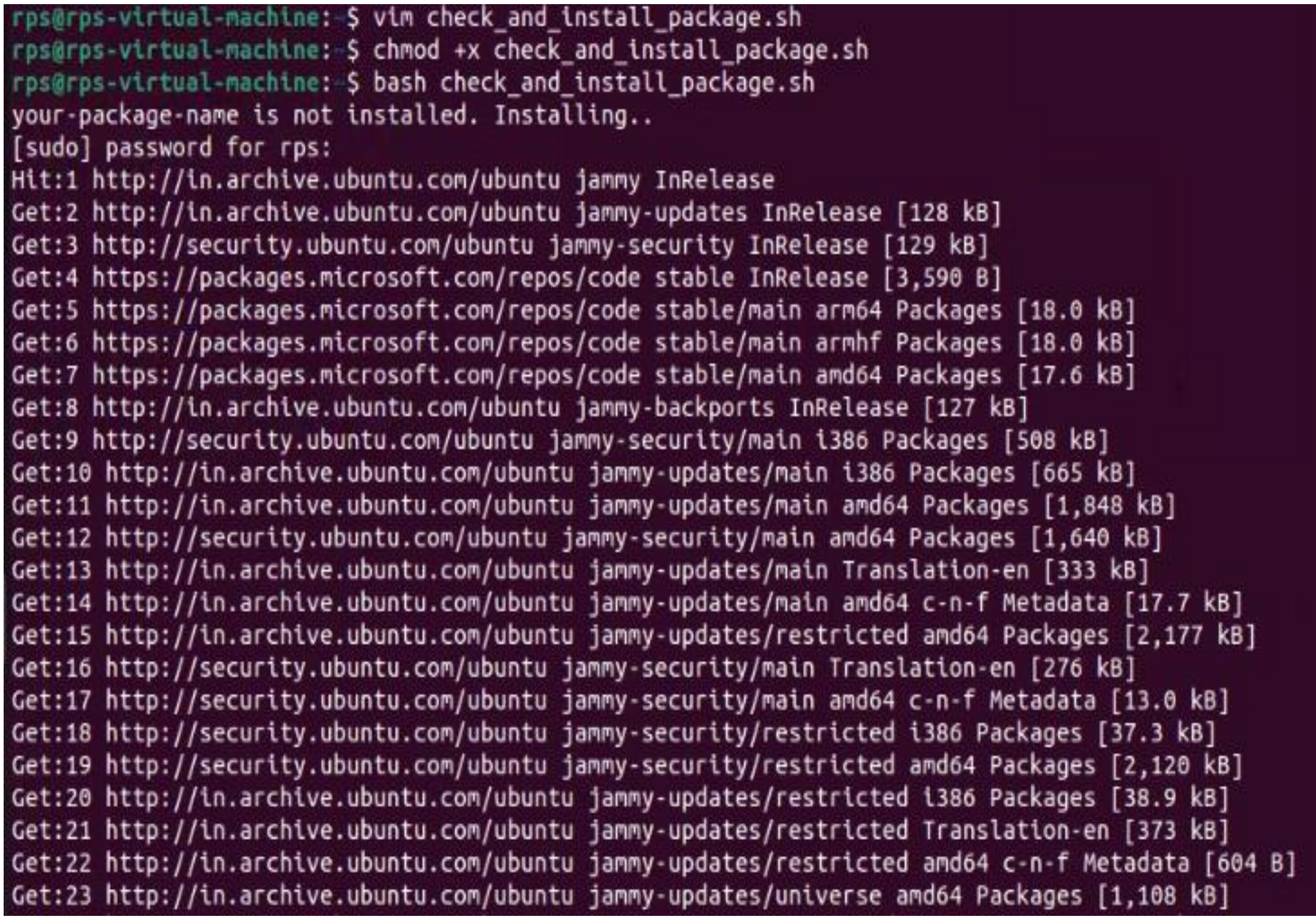
**Create a script that takes a text file as input and replaces all occurrences of a specific word with another word.**



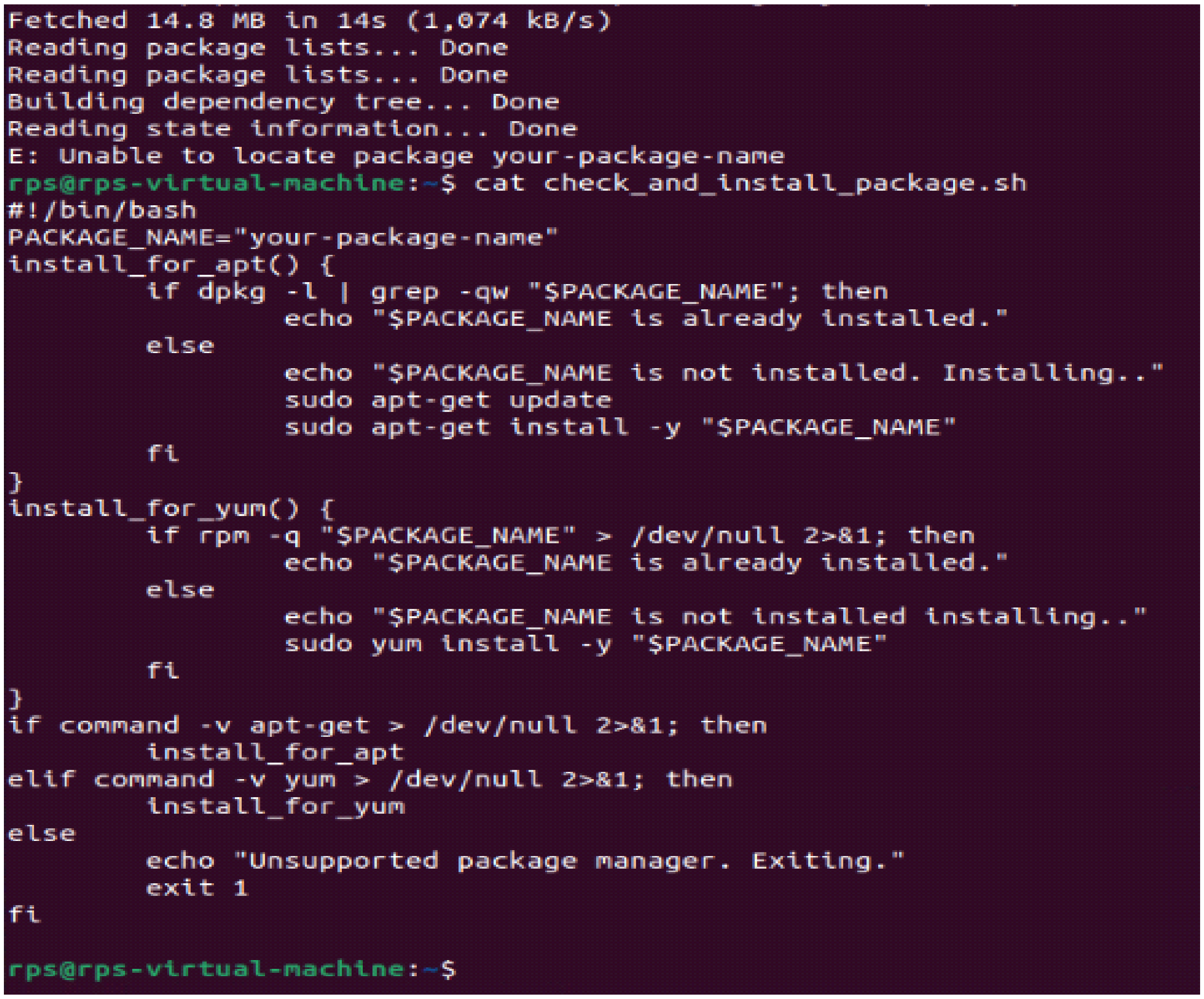
```
rps@rps-virtual-machine:~$ vim replace_word.sh
rps@rps-virtual-machine:~$ bash replace_word.sh /home/rps/example.txt old_word new_word
Replaced all occurences of 'old_word' with 'new_word' in /home/rps/example.txt
rps@rps-virtual-machine:~$ cat replace_word.sh
#!/bin/bash
if [ "$#" -ne 3 ]; then
        echo "Usage: $0 <file_path> <word_TO_replace> <replacement_word>"
        exit 1
fi
file_path=$1
word_to_replace=$2
replacement_word=$3
temp_file=$(mktemp)
sed "s/$word_to_replace/$replacement_word/g" "$file_path" > "$temp_file"
mv "$temp_file" "$file_path"
echo "Replaced all occurences of '$word_to_replace' with '$replacement_word' in $file_path"
rps@rps-virtual-machine:~$
```

**4. System Administration:**

**Write a script that checks if a specific package is installed and, if not, installs it using the appropriate package manager (e.g., apt-get, yum).**

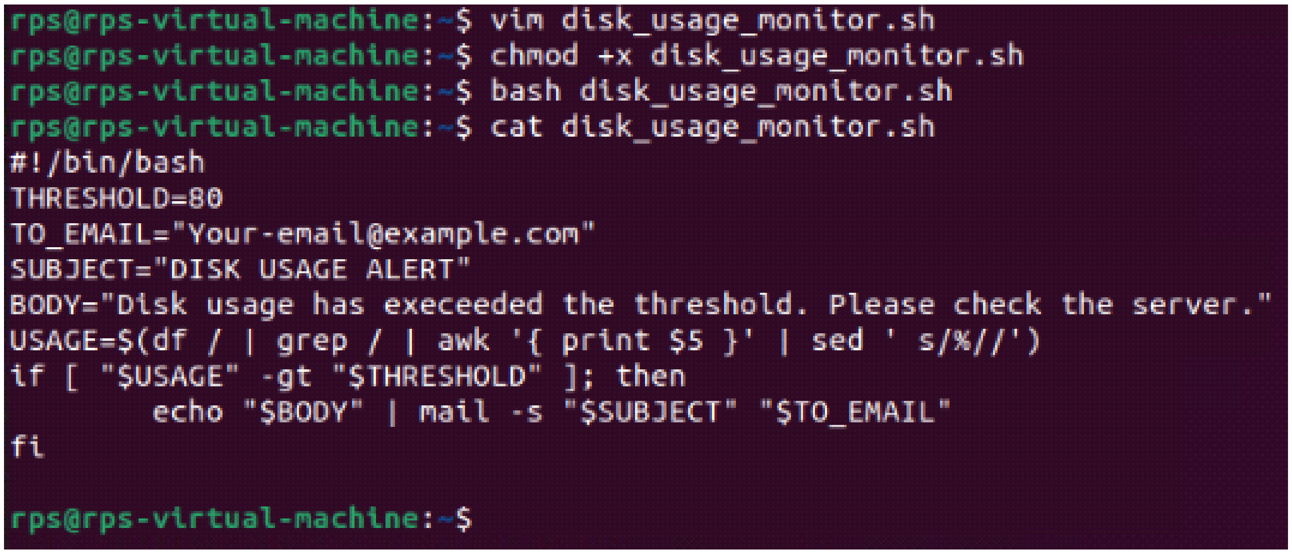


```
rps@rps-virtual-machine:~$ vim check_and_install_package.sh
rps@rps-virtual-machine:~$ chmod +x check_and_install_package.sh
rps@rps-virtual-machine:~$ bash check_and_install_package.sh
your-package-name is not installed. Installing..
[sudo] password for rps:
Hit:1 http://in.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://in.archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]
Get:3 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Get:4 https://packages.microsoft.com/repos/code stable InRelease [3,590 B]
Get:5 https://packages.microsoft.com/repos/code stable/main arm64 Packages [18.0 kB]
Get:6 https://packages.microsoft.com/repos/code stable/main armhf Packages [18.0 kB]
Get:7 https://packages.microsoft.com/repos/code stable/main amd64 Packages [17.6 kB]
Get:8 http://in.archive.ubuntu.com/ubuntu jammy-backports InRelease [127 kB]
Get:9 http://security.ubuntu.com/ubuntu jammy-security/main i386 Packages [508 kB]
Get:10 http://in.archive.ubuntu.com/ubuntu jammy-updates/main i386 Packages [665 kB]
Get:11 http://in.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [1,848 kB]
Get:12 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [1,640 kB]
Get:13 http://in.archive.ubuntu.com/ubuntu jammy-updates/main Translation-en [333 kB]
Get:14 http://in.archive.ubuntu.com/ubuntu jammy-updates/main amd64 c-n-f Metadata [17.7 kB]
Get:15 http://in.archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 Packages [2,177 kB]
Get:16 http://security.ubuntu.com/ubuntu jammy-security/main Translation-en [276 kB]
Get:17 http://security.ubuntu.com/ubuntu jammy-security/main amd64 c-n-f Metadata [13.0 kB]
Get:18 http://security.ubuntu.com/ubuntu jammy-security/restricted i386 Packages [37.3 kB]
Get:19 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 Packages [2,120 kB]
Get:20 http://in.archive.ubuntu.com/ubuntu jammy-updates/restricted i386 Packages [38.9 kB]
Get:21 http://in.archive.ubuntu.com/ubuntu jammy-updates/restricted Translation-en [373 kB]
Get:22 http://in.archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 c-n-f Metadata [604 B]
Get:23 http://in.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [1,108 kB]
```



```
Fetched 14.8 MB in 14s (1,074 kB/s)
Reading package lists... Done
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
E: Unable to locate package your-package-name
rps@rps-virtual-machine:~$ cat check_and_install_package.sh
#!/bin/bash
PACKAGE_NAME="your-package-name"
install_for_apt() {
        if dpkg -l | grep -qw "$PACKAGE_NAME"; then
                echo "$PACKAGE_NAME is already installed."
        else
                echo "$PACKAGE_NAME is not installed. Installing.."
                sudo apt-get update
                sudo apt-get install -y "$PACKAGE_NAME"
        fi
}
install_for_yum() {
        if rpm -q "$PACKAGE_NAME" > /dev/null 2>&1; then
                echo "$PACKAGE_NAME is already installed."
        else
                echo "$PACKAGE_NAME is not installed installing.."
                sudo yum install -y "$PACKAGE_NAME"
        fi
}
if command -v apt-get > /dev/null 2>&1; then
        install_for_apt
elif command -v yum > /dev/null 2>&1; then
        install_for_yum
else
        echo "Unsupported package manager. Exiting."
        exit 1
fi

rps@rps-virtual-machine:~$
```

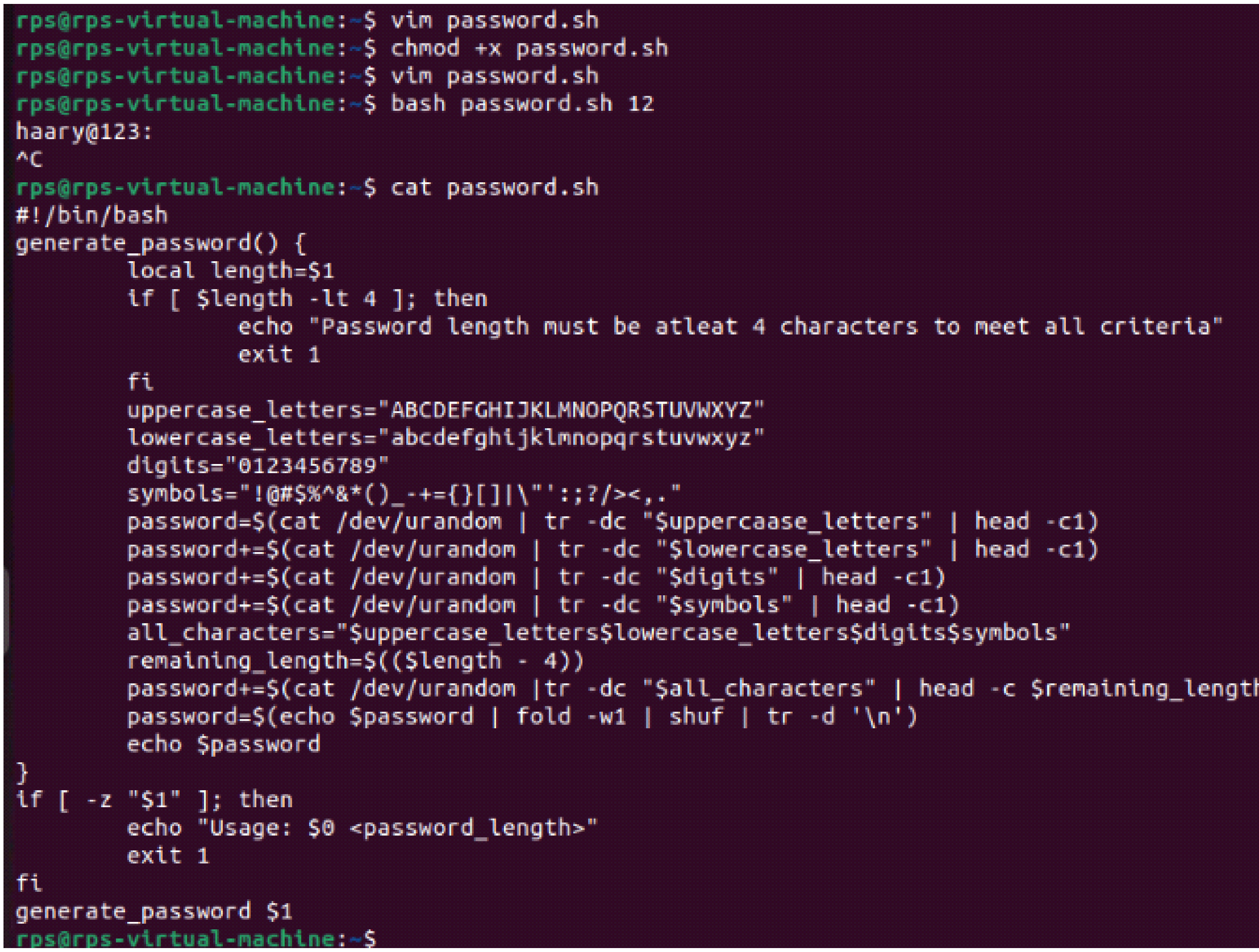
**Create a script that monitors disk usage and sends an email notification if it exceeds a certain threshold**.



```
rps@rps-virtual-machine:~$ vim disk_usage_monitor.sh
rps@rps-virtual-machine:~$ chmod +x disk_usage_monitor.sh
rps@rps-virtual-machine:~$ bash disk_usage_monitor.sh
rps@rps-virtual-machine:~$ cat disk_usage_monitor.sh
#!/bin/bash
THRESHOLD=80
TO_EMAIL="Your-email@example.com"
SUBJECT="DISK USAGE ALERT"
BODY="Disk usage has execeeded the threshold. Please check the server."
USAGE=$(df / | grep / | awk '{ print $5 }' | sed ' s/%//')
if [ "$USAGE" -gt "$THRESHOLD" ]; then
        echo "$BODY" | mail -s "$SUBJECT" "$TO_EMAIL"
fi

rps@rps-virtual-machine:~$
```

**5. Data Manupulation**

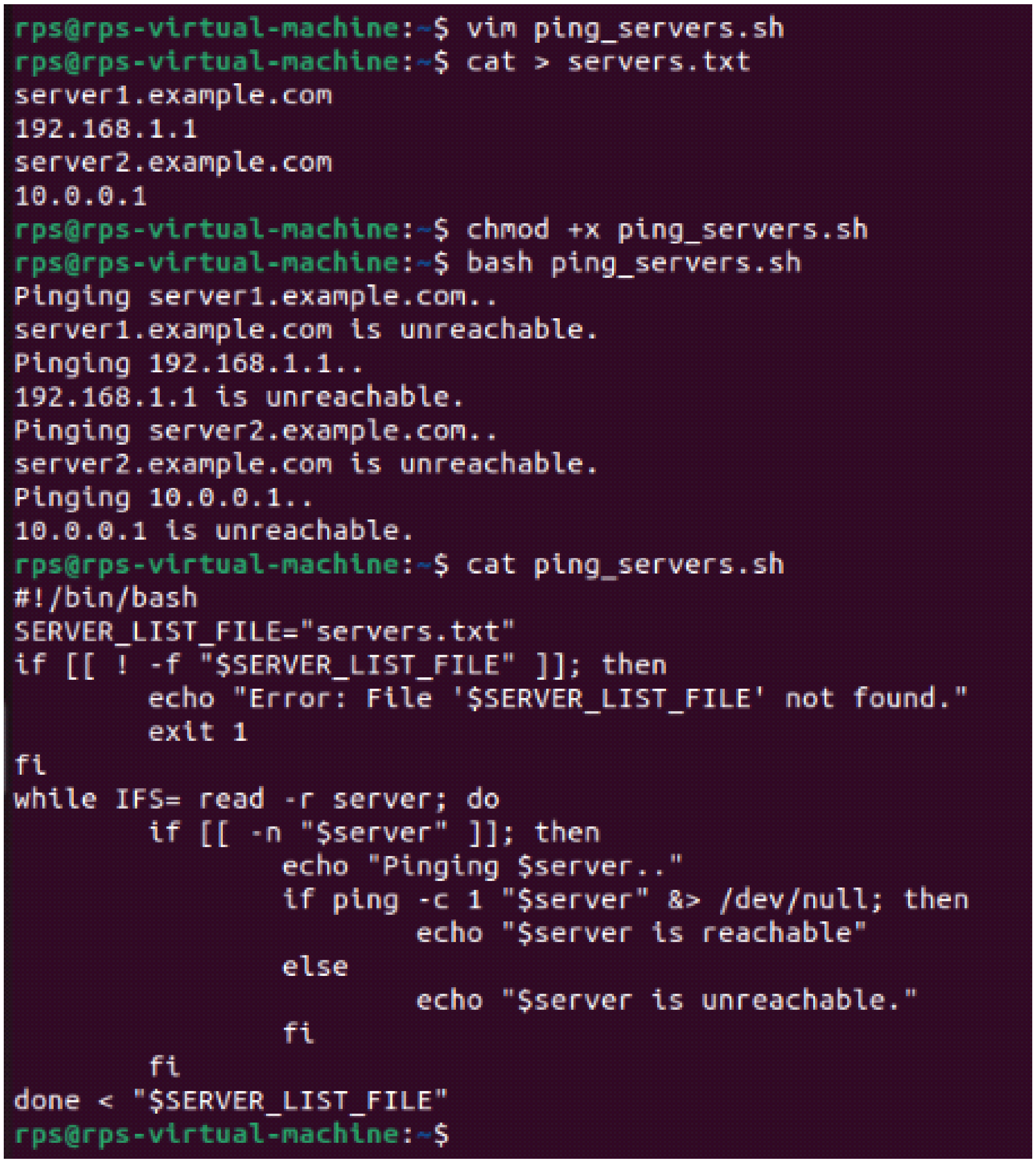
**Create a script that generates a random password of a specified length, meeting certain criteria like uppercase, lowercase, numbers, and symbols.**



```
rps@rps-virtual-machine:~$ vim password.sh
rps@rps-virtual-machine:~$ chmod +x password.sh
rps@rps-virtual-machine:~$ vim password.sh
rps@rps-virtual-machine:~$ bash password.sh 12
haary@123:
^C
rps@rps-virtual-machine:~$ cat password.sh
#!/bin/bash
generate_password() {
        local length=$1
        if [ $length -lt 4 ]; then
                echo "Password length must be atleat 4 characters to meet all criteria"
                exit 1
        fi
        uppercase_letters="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
        lowercase_letters="abcdefghijklmnopqrstuvwxyz"
        digits="0123456789"
        symbols="!@#$%^&*()_-+={}[]|\"':;?/><,."
        password=$(cat /dev/urandom | tr -dc "$uppercaase_letters" | head -c1)
        password+=$(cat /dev/urandom | tr -dc "$lowercase_letters" | head -c1)
        password+=$(cat /dev/urandom | tr -dc "$digits" | head -c1)
        password+=$(cat /dev/urandom | tr -dc "$symbols" | head -c1)
        all_characters="$uppercase_letters$lowercase_letters$digits$symbols"
        remaining_length=$(($length - 4))
        password+=$(cat /dev/urandom |tr -dc "$all_characters" | head -c $remaining_lengt
        password=$(echo $password | fold -w1 | shuf | tr -d '\n')
        echo $password
}
if [ -z "$1" ]; then
        echo "Usage: $0 <password_length>"
        exit 1
fi
generate_password $1
rps@rps-virtual-machine:~$
```

**6. Network Operations:**

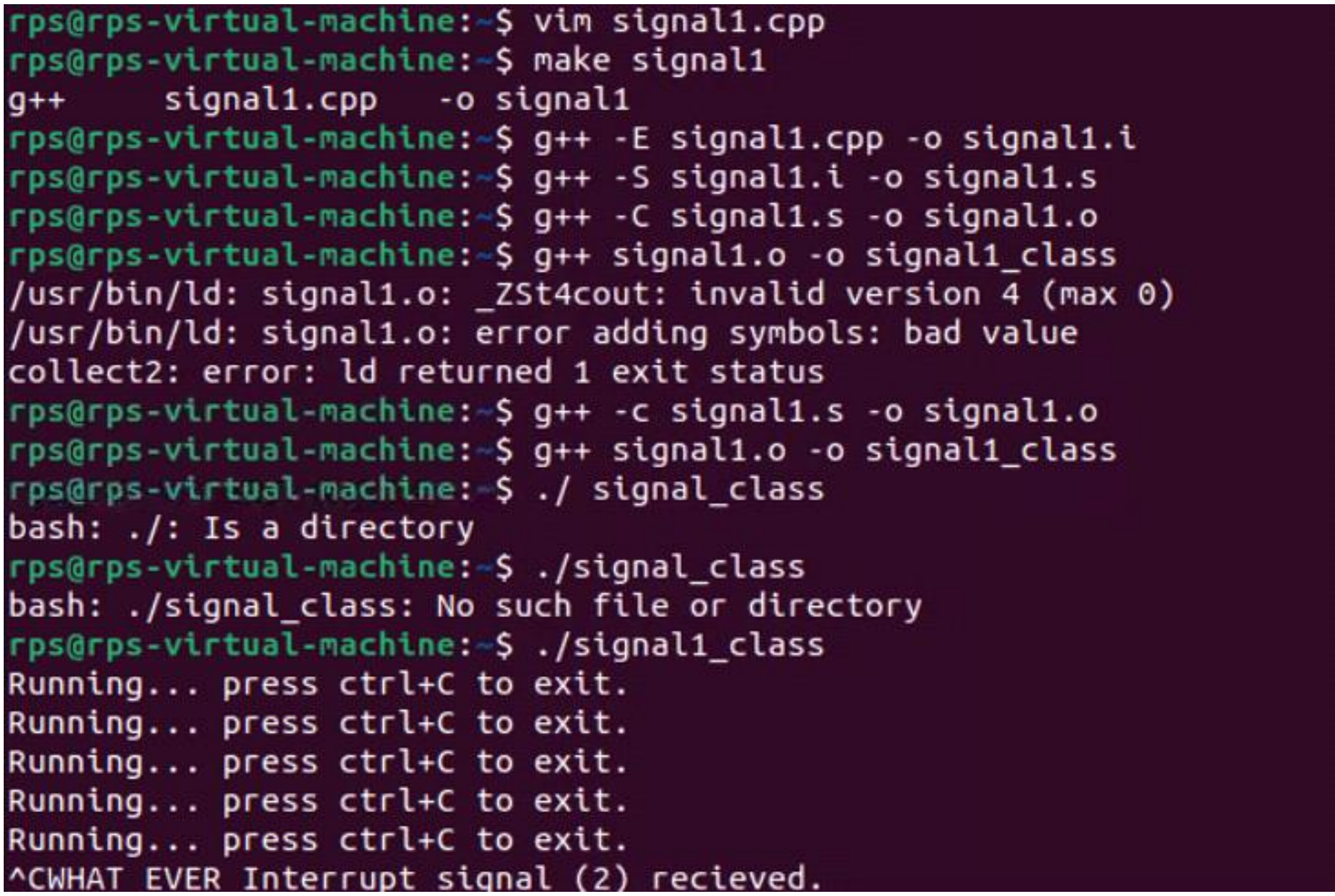
**Write a script that pings a list of servers and reports if any are unreachable.**



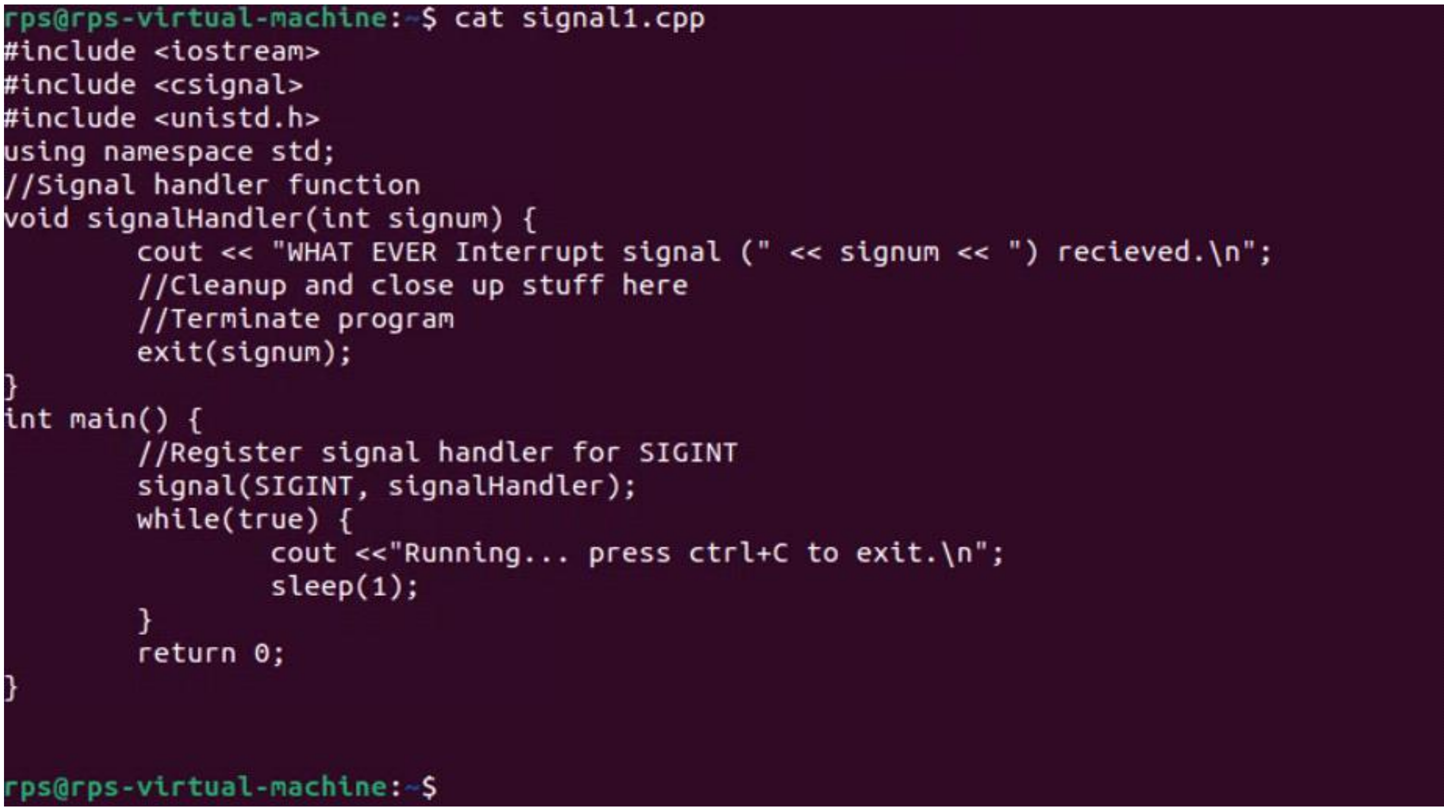
```
rps@rps-virtual-machine:~$ vim ping_servers.sh
rps@rps-virtual-machine:~$ cat > servers.txt
server1.example.com
192.168.1.1
server2.example.com
10.0.0.1
rps@rps-virtual-machine:~$ chmod +x ping_servers.sh
rps@rps-virtual-machine:~$ bash ping_servers.sh
Pinging server1.example.com..
server1.example.com is unreachable.
Pinging 192.168.1.1..
192.168.1.1 is unreachable.
Pinging server2.example.com..
server2.example.com is unreachable.
Pinging 10.0.0.1..
10.0.0.1 is unreachable.
rps@rps-virtual-machine:~$ cat ping_servers.sh
#!/bin/bash
SERVER_LIST_FILE="servers.txt"
if [[ ! -f "$SERVER_LIST_FILE" ]]; then
        echo "Error: File '$SERVER_LIST_FILE' not found."
        exit 1
fi
while IFS= read -r server; do
        if [[ -n "$server" ]]; then
                echo "Pinging $server.."
                if ping -c 1 "$server" &> /dev/null; then
                        echo "$server is reachable"
                else
                        echo "$server is unreachable."
                fi
        fi
done < "$SERVER_LIST_FILE"
rps@rps-virtual-machine:~$
```

**SIGNALS :-**

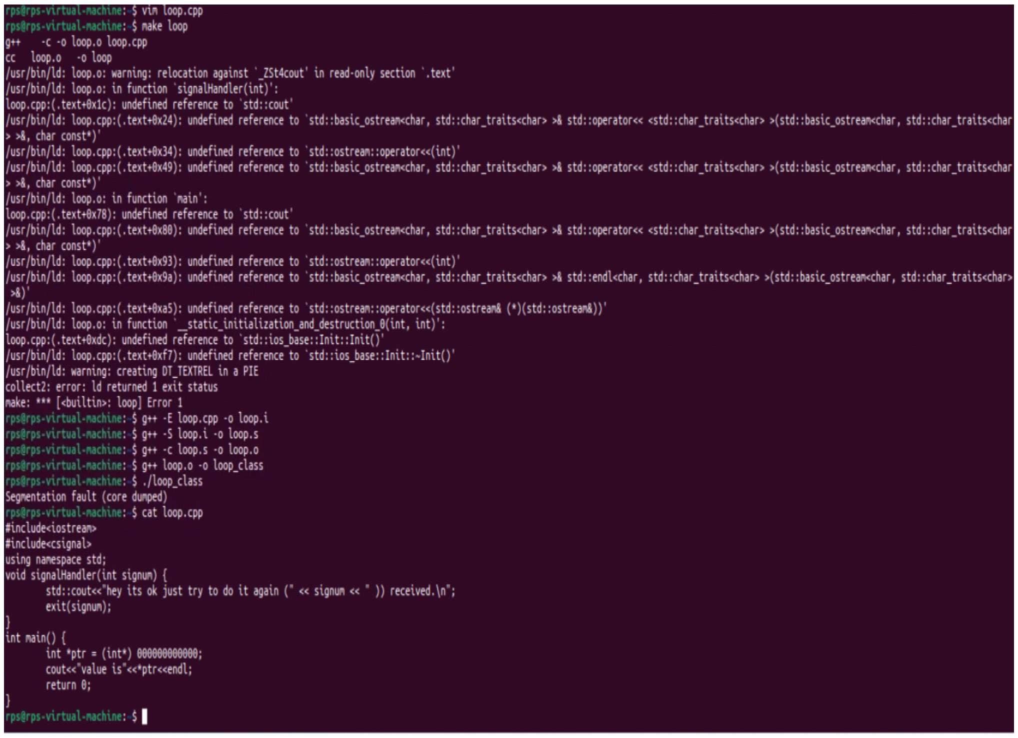
**SIGNAL PROGRAM :-**



```
rps@rps-virtual-machine:~$ vim signal1.cpp
rps@rps-virtual-machine:~$ make signal1
g++     signal1.cpp   -o signal1
rps@rps-virtual-machine:~$ g++ -E signal1.cpp -o signal1.i
rps@rps-virtual-machine:~$ g++ -S signal1.i -o signal1.s
rps@rps-virtual-machine:~$ g++ -C signal1.s -o signal1.o
rps@rps-virtual-machine:~$ g++ signal1.o -o signal1_class
/usr/bin/ld: signal1.o: _ZSt4cout: invalid version 4 (max 0)
/usr/bin/ld: signal1.o: error adding symbols: bad value
collect2: error: ld returned 1 exit status
rps@rps-virtual-machine:~$ g++ -c signal1.s -o signal1.o
rps@rps-virtual-machine:~$ g++ signal1.o -o signal1_class
rps@rps-virtual-machine:~$ ./ signal_class
bash: ./: Is a directory
rps@rps-virtual-machine:~$ ./signal_class
bash: ./signal_class: No such file or directory
rps@rps-virtual-machine:~$ ./signal1_class
Running... press ctrl+C to exit.
Running... press ctrl+C to exit.
Running... press ctrl+C to exit.
Running... press ctrl+C to exit.
Running... press ctrl+C to exit.
^CWHAT EVER Interrupt signal (2) recieved.
```

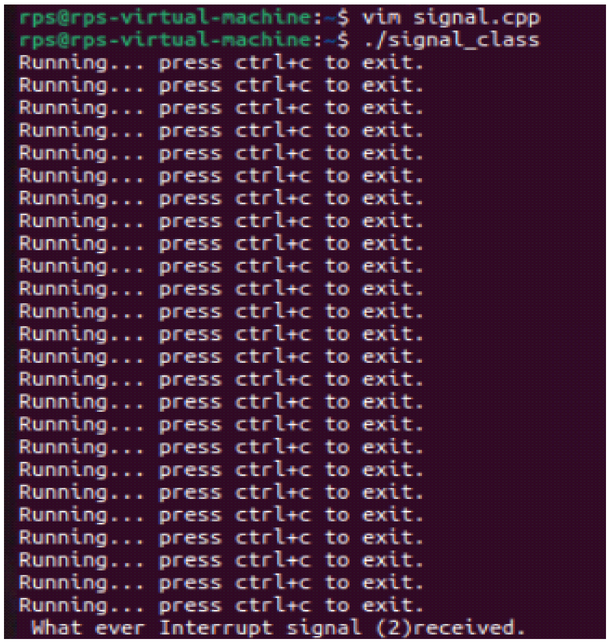


```
rps@rps-virtual-machine:~$ cat signal1.cpp
#include <iostream>
#include <csignal>
#include <unistd.h>
using namespace std;
//Signal handler function
void signalHandler(int signum) {
        cout << "WHAT EVER Interrupt signal (" << signum << ") recieved.\n";
        //Cleanup and close up stuff here
        //Terminate program
        exit(signum);
}
int main() {
        //Register signal handler for SIGINT
        signal(SIGINT, signalHandler);
        while(true) {
                cout <<"Running... press ctrl+C to exit.\n";
                sleep(1);
        }
        return 0;
}

rps@rps-virtual-machine:~$
```



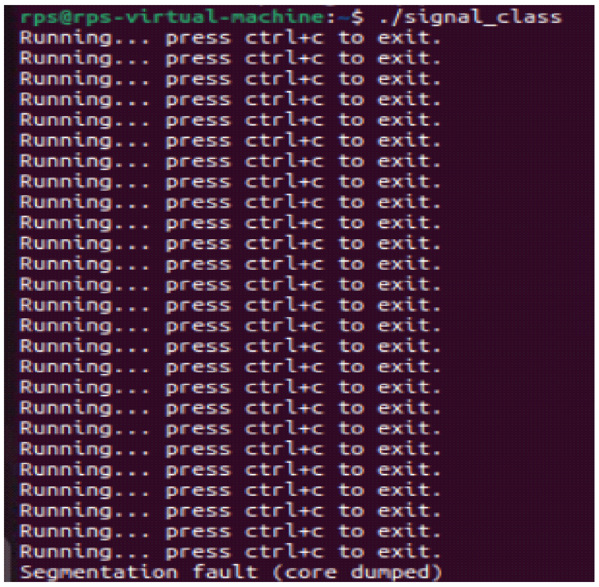
```
rps@rps-virtual-machine:~$ vim loop.cpp
rps@rps-virtual-machine:~$ make loop
g++     -c -o loop.o loop.cpp
cc   loop.o   -o loop
/usr/bin/ld: loop.o: warning: relocation against `_ZSt4cout' in read-only section `.text'
/usr/bin/ld: loop.o: in function `signalHandler(int)':
loop.cpp:(.text+0x1c): undefined reference to `std::cout'
/usr/bin/ld: loop.cpp:(.text+0x24): undefined reference to `std::basic_ostream<char, std::char_traits<char> >& std::operator<< <std::char_traits<char> >(std::basic_ostream<char, std::char_traits<char
> >&, char const*)'
/usr/bin/ld: loop.cpp:(.text+0x34): undefined reference to `std::ostream::operator<<(int)'
/usr/bin/ld: loop.cpp:(.text+0x49): undefined reference to `std::basic_ostream<char, std::char_traits<char> >& std::operator<< <std::char_traits<char> >(std::basic_ostream<char, std::char_traits<char
> >&, char const*)'
/usr/bin/ld: loop.o: in function `main':
loop.cpp:(.text+0x78): undefined reference to `std::cout'
/usr/bin/ld: loop.cpp:(.text+0x80): undefined reference to `std::basic_ostream<char, std::char_traits<char> >& std::operator<< <std::char_traits<char> >(std::basic_ostream<char, std::char_traits<char
> >&, char const*)'
/usr/bin/ld: loop.cpp:(.text+0x93): undefined reference to `std::ostream::operator<<(int)'
/usr/bin/ld: loop.cpp:(.text+0x9a): undefined reference to `std::basic_ostream<char, std::char_traits<char> >& std::endl<char, std::char_traits<char> >(std::basic_ostream<char, std::char_traits<char>
 >&)'
/usr/bin/ld: loop.cpp:(.text+0xa5): undefined reference to `std::ostream::operator<<(std::ostream& (*)(std::ostream&))'
/usr/bin/ld: loop.o: in function `__static_initialization_and_destruction_0(int, int)':
loop.cpp:(.text+0xdc): undefined reference to `std::ios_base::Init::Init()'
/usr/bin/ld: loop.cpp:(.text+0xf7): undefined reference to `std::ios_base::Init::~Init()'
/usr/bin/ld: warning: creating DT_TEXTREL in a PIE
collect2: error: ld returned 1 exit status
make: *** [<builtin>: loop] Error 1
rps@rps-virtual-machine:~$ g++ -E loop.cpp -o loop.i
rps@rps-virtual-machine:~$ g++ -S loop.i -o loop.s
rps@rps-virtual-machine:~$ g++ -c loop.s -o loop.o
rps@rps-virtual-machine:~$ g++ loop.o -o loop_class
rps@rps-virtual-machine:~$ ./loop_class
Segmentation fault (core dumped)
rps@rps-virtual-machine:~$ cat loop.cpp
#include<iostream>
#include<csignal>
using namespace std;
void signalHandler(int signum) {
        std::cout<<"hey its ok just try to do it again (" << signum << " )) received.\n";
        exit(signum);
}
int main() {
        int *ptr = (int*) 000000000000;
        cout<<"value is"<<*ptr<<endl;
        return 0;
}
rps@rps-virtual-machine:~$
```

**USING KILL SIGNINT :-**

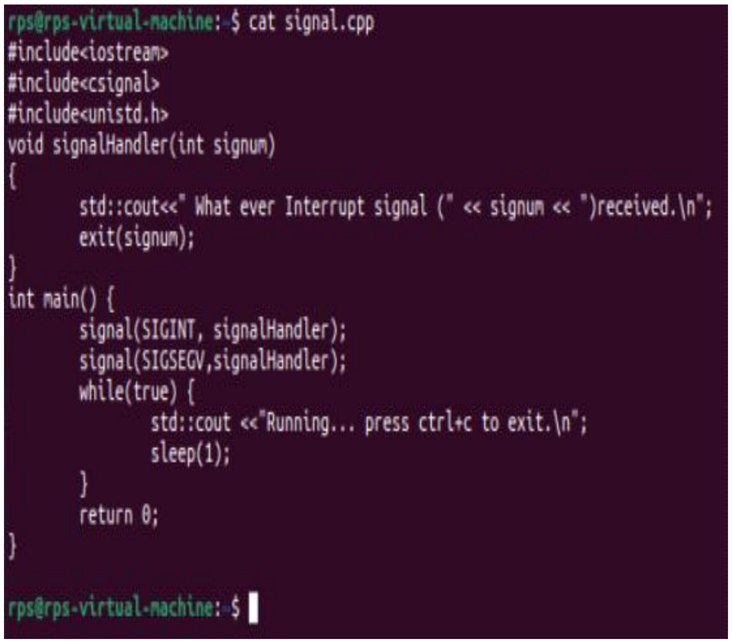


```
rps@rps-virtual-machine:~$ vim signal.cpp
rps@rps-virtual-machine:~$ ./signal_class
Running... press ctrl+c to exit.
Running... press ctrl+c to exit.
Running... press ctrl+c to exit.
Running... press ctrl+c to exit.
Running... press ctrl+c to exit.
Running... press ctrl+c to exit.
Running... press ctrl+c to exit.
Running... press ctrl+c to exit.
Running... press ctrl+c to exit.
Running... press ctrl+c to exit.
Running... press ctrl+c to exit.
Running... press ctrl+c to exit.
Running... press ctrl+c to exit.
Running... press ctrl+c to exit.
Running... press ctrl+c to exit.
Running... press ctrl+c to exit.
Running... press ctrl+c to exit.
Running... press ctrl+c to exit.
Running... press ctrl+c to exit.
Running... press ctrl+c to exit.
Running... press ctrl+c to exit.
Running... press ctrl+c to exit.
Running... press ctrl+c to exit.
Running... press ctrl+c to exit.
Running... press ctrl+c to exit.
 What ever Interrupt signal (2)received.
```

**USING KILL SIGSEGV :-**

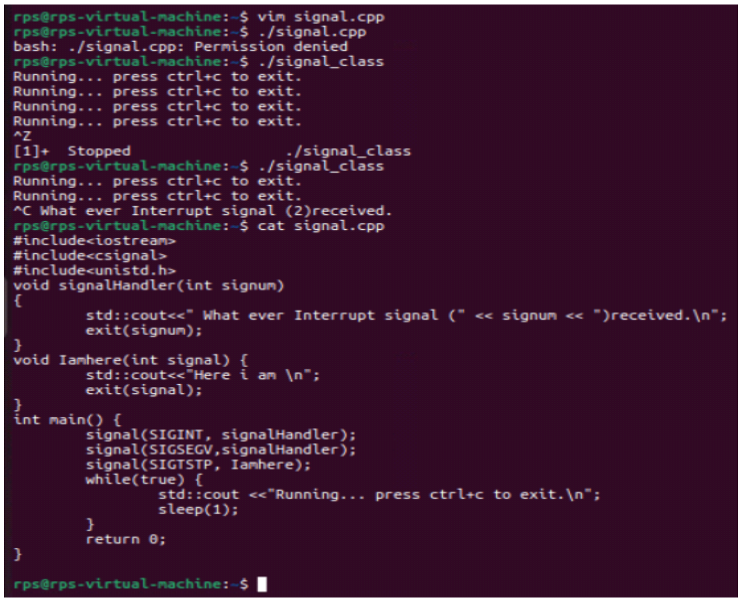


```
rps@rps-virtual-machine:~$ ./signal_class
Running... press ctrl+c to exit.
Running... press ctrl+c to exit.
Running... press ctrl+c to exit.
Running... press ctrl+c to exit.
Running... press ctrl+c to exit.
Running... press ctrl+c to exit.
Running... press ctrl+c to exit.
Running... press ctrl+c to exit.
Running... press ctrl+c to exit.
Running... press ctrl+c to exit.
Running... press ctrl+c to exit.
Running... press ctrl+c to exit.
Running... press ctrl+c to exit.
Running... press ctrl+c to exit.
Running... press ctrl+c to exit.
Running... press ctrl+c to exit.
Running... press ctrl+c to exit.
Running... press ctrl+c to exit.
Running... press ctrl+c to exit.
Running... press ctrl+c to exit.
Running... press ctrl+c to exit.
Running... press ctrl+c to exit.
Running... press ctrl+c to exit.
Running... press ctrl+c to exit.
Running... press ctrl+c to exit.
Running... press ctrl+c to exit.
Segmentation fault (core dumped)
```



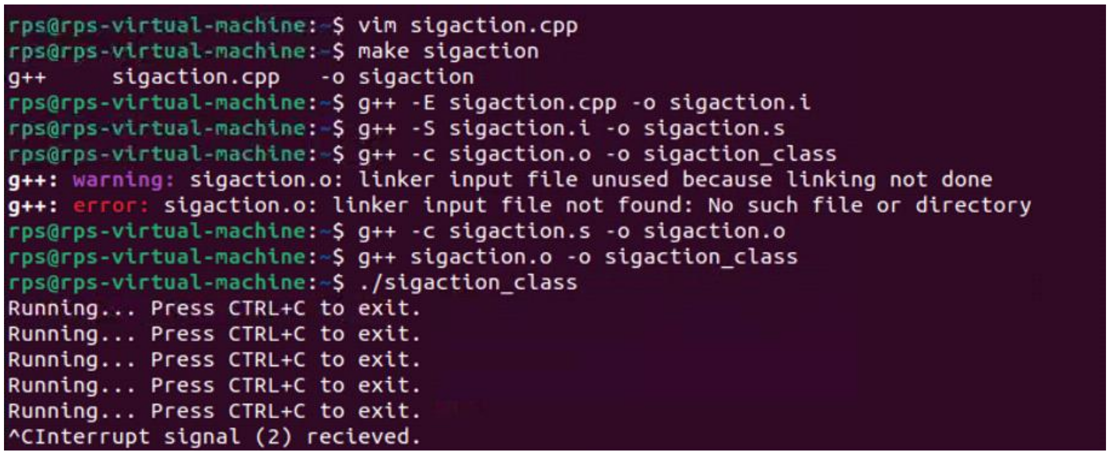
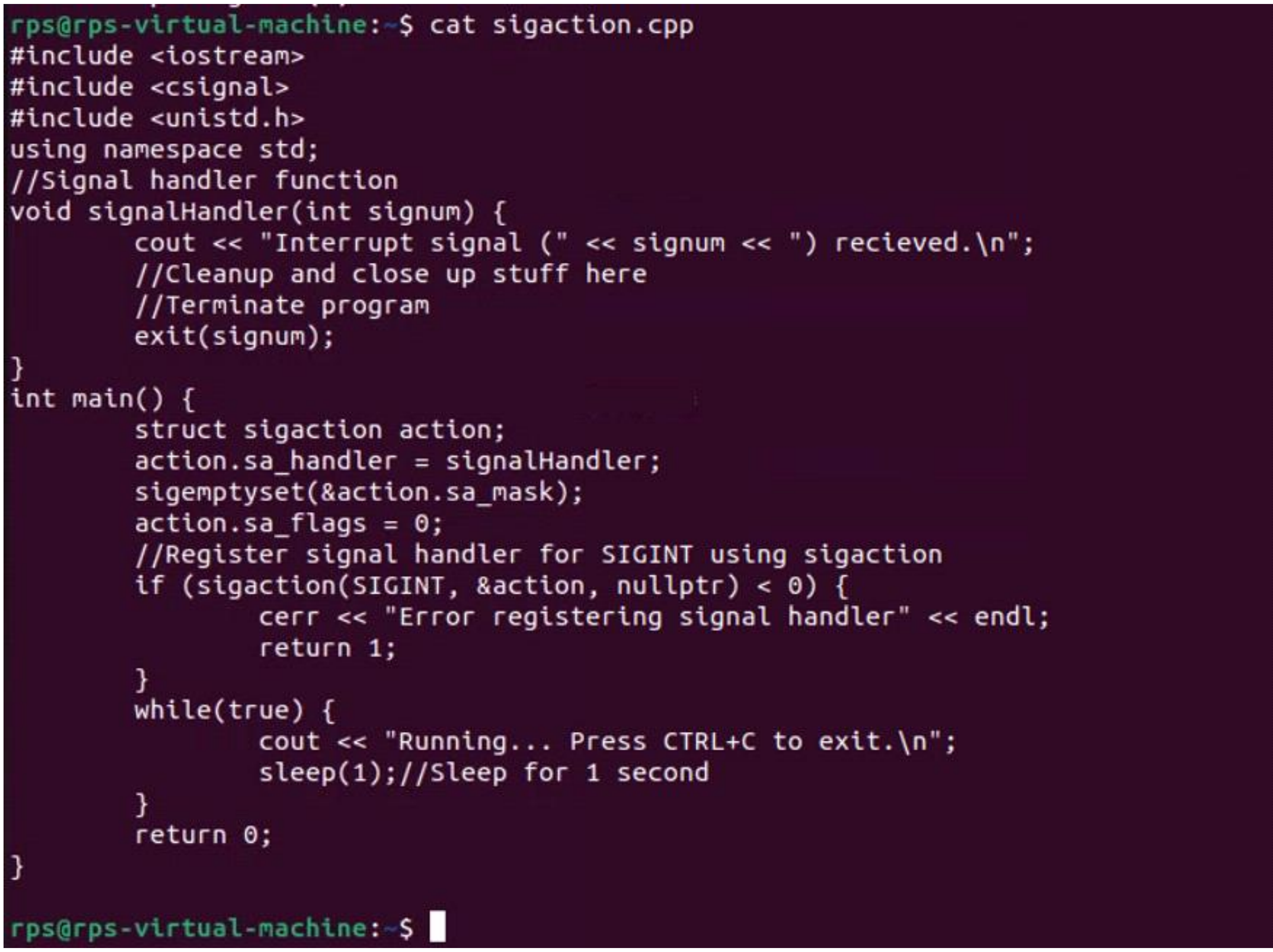
```
rps@rps-virtual-machine:~$ cat signal.cpp
#include<iostream>
#include<csignal>
#include<unistd.h>
void signalHandler(int signum)
{
        std::cout<<" What ever Interrupt signal (" << signum << ")received.\n";
        exit(signum);
}
int main() {
        signal(SIGINT, signalHandler);
        signal(SIGSEGV,signalHandler);
        while(true) {
                std::cout <<"Running... press ctrl+c to exit.\n";
                sleep(1);
        }
        return 0;
}

rps@rps-virtual-machine:~$
```

## Using ctrl+c and ctrl+z:

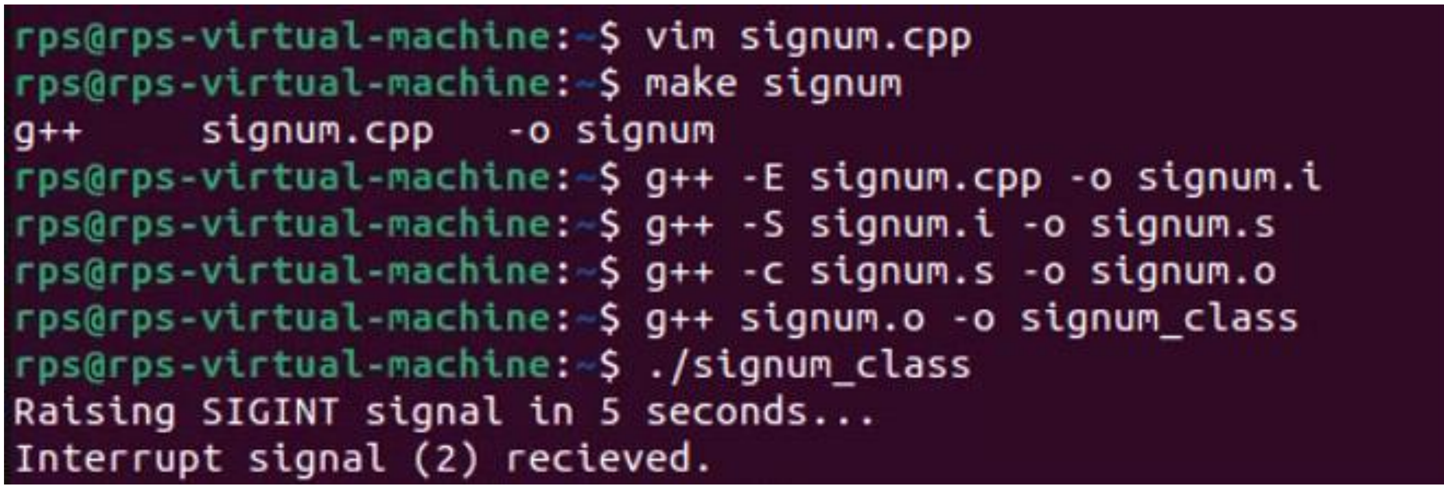


```
rps@rps-virtual-machine:~$ vim signal.cpp
rps@rps-virtual-machine:~$ ./signal.cpp
bash: ./signal.cpp: Permission denied
rps@rps-virtual-machine:~$ ./signal_class
Running... press ctrl+c to exit.
Running... press ctrl+c to exit.
Running... press ctrl+c to exit.
Running... press ctrl+c to exit.
^Z
[1]+  Stopped                 ./signal_class
rps@rps-virtual-machine:~$ ./signal_class
Running... press ctrl+c to exit.
Running... press ctrl+c to exit.
^C What ever Interrupt signal (2)received.
rps@rps-virtual-machine:~$ cat signal.cpp
#include<iostream>
#include<csignal>
#include<unistd.h>
void signalHandler(int signum)
{
        std::cout<<" What ever Interrupt signal (" << signum << ")received.\n";
        exit(signum);
}
void Iamhere(int signal) {
        std::cout<<"Here i am \n";
        exit(signal);
}
int main() {
        signal(SIGINT, signalHandler);
        signal(SIGSEGV,signalHandler);
        signal(SIGTSTP, Iamhere);
        while(true) {
                std::cout <<"Running... press ctrl+c to exit.\n";
                sleep(1);
        }
        return 0;
}

rps@rps-virtual-machine:~$
```

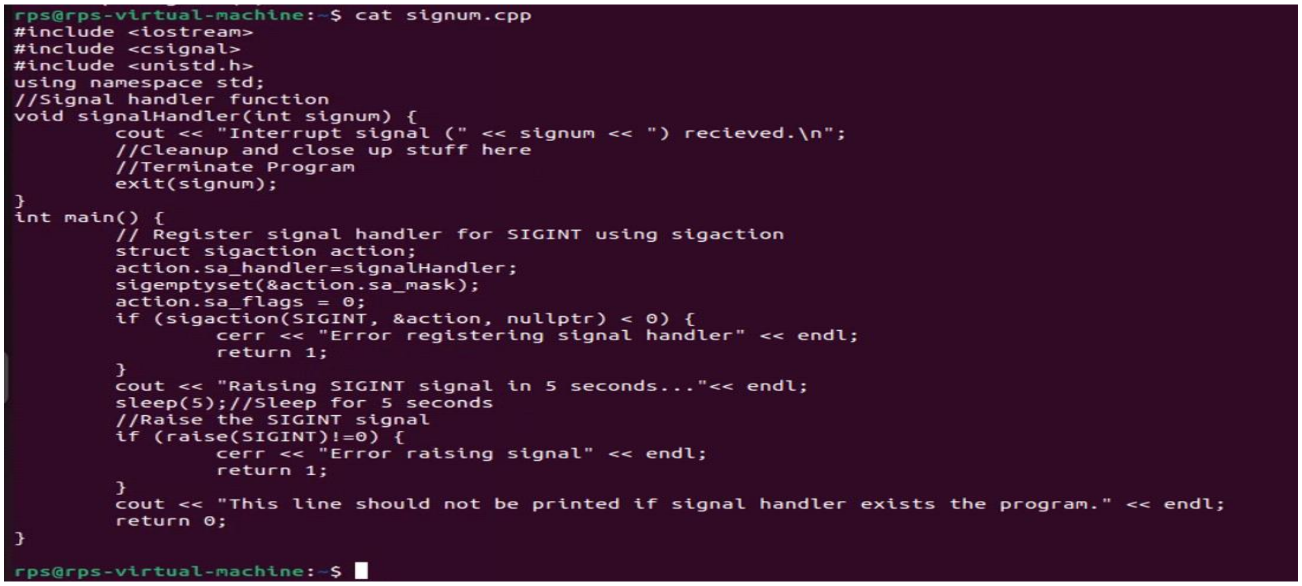
## SIGACTION PROGRAM :-


```
rps@rps-virtual-machine:~$ vim sigaction.cpp
rps@rps-virtual-machine:~$ make sigaction
g++     sigaction.cpp   -o sigaction
rps@rps-virtual-machine:~$ g++ -E sigaction.cpp -o sigaction.i
rps@rps-virtual-machine:~$ g++ -S sigaction.i -o sigaction.s
rps@rps-virtual-machine:~$ g++ -c sigaction.o -o sigaction_class
g++: warning: sigaction.o: linker input file unused because linking not done
g++: error: sigaction.o: linker input file not found: No such file or directory
rps@rps-virtual-machine:~$ g++ -c sigaction.s -o sigaction.o
rps@rps-virtual-machine:~$ g++ sigaction.o -o sigaction_class
rps@rps-virtual-machine:~$ ./sigaction_class
Running... Press CTRL+C to exit.
Running... Press CTRL+C to exit.
Running... Press CTRL+C to exit.
Running... Press CTRL+C to exit.
Running... Press CTRL+C to exit.
^CInterrupt signal (2) recieved.
```

```
rps@rps-virtual-machine:~$ cat sigaction.cpp
#include <iostream>
#include <csignal>
#include <unistd.h>
using namespace std;
//Signal handler function
void signalHandler(int signum) {
        cout << "Interrupt signal (" << signum << ") recieved.\n";
        //Cleanup and close up stuff here
        //Terminate program
        exit(signum);
}
int main() {
        struct sigaction action;
        action.sa_handler = signalHandler;
        sigemptyset(&action.sa_mask);
        action.sa_flags = 0;
        //Register signal handler for SIGINT using sigaction
        if (sigaction(SIGINT, &action, nullptr) < 0) {
                cerr << "Error registering signal handler" << endl;
                return 1;
        }
        while(true) {
                cout << "Running... Press CTRL+C to exit.\n";
                sleep(1);//Sleep for 1 second
        }
        return 0;
}

rps@rps-virtual-machine:~$ 
```

**SIGNUM PROGRAM :-**

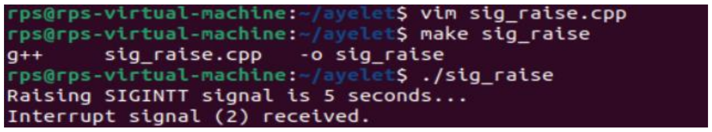


```
rps@rps-virtual-machine:~$ vim signum.cpp
rps@rps-virtual-machine:~$ make signum
g++     signum.cpp   -o signum
rps@rps-virtual-machine:~$ g++ -E signum.cpp -o signum.i
rps@rps-virtual-machine:~$ g++ -S signum.i -o signum.s
rps@rps-virtual-machine:~$ g++ -c signum.s -o signum.o
rps@rps-virtual-machine:~$ g++ signum.o -o signum_class
rps@rps-virtual-machine:~$ ./signum_class
Raising SIGINT signal in 5 seconds...
Interrupt signal (2) recieved.
```

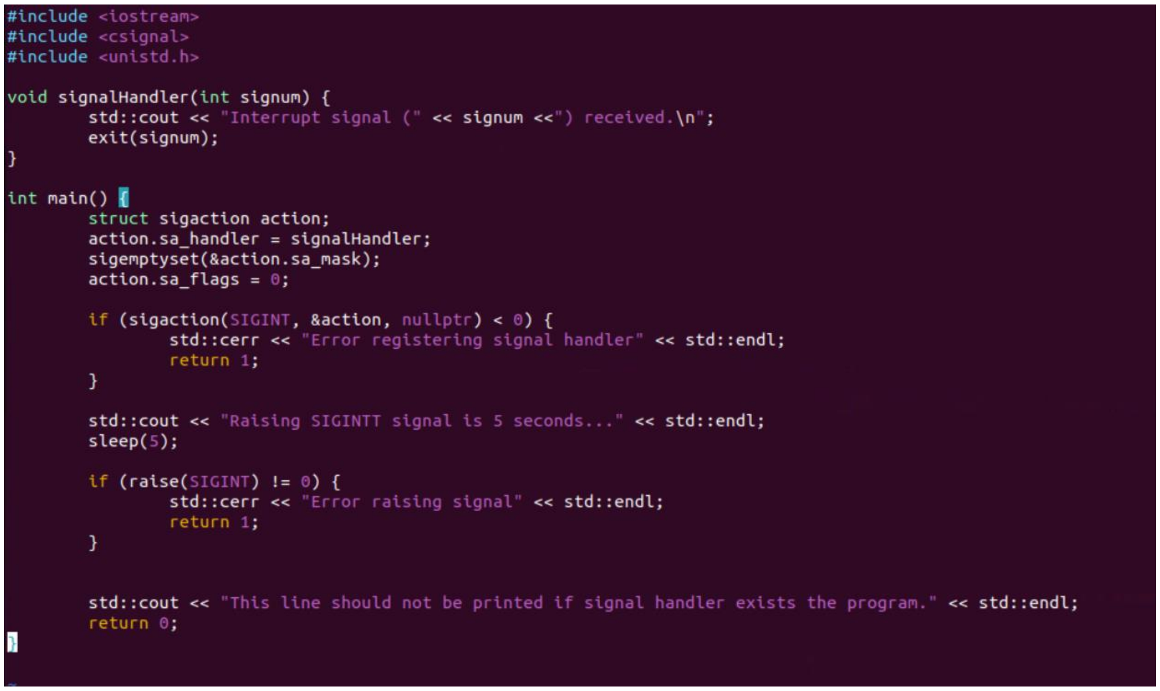


```
rps@rps-virtual-machine:~$ cat signum.cpp
#include <iostream>
#include <csignal>
#include <unistd.h>
using namespace std;
//Signal handler function
void signalHandler(int signum) {
        cout << "Interrupt signal (" << signum << ") recieved.\n";
        //Cleanup and close up stuff here
        //Terminate Program
        exit(signum);
}
int main() {
        // Register signal handler for SIGINT using sigaction
        struct sigaction action;
        action.sa_handler=signalHandler;
        sigemptyset(&action.sa_mask);
        action.sa_flags = 0;
        if (sigaction(SIGINT, &action, nullptr) < 0) {
                cerr << "Error registering signal handler" << endl;
                return 1;
        }
        cout << "Raising SIGINT signal in 5 seconds..."<< endl;
        sleep(5);//Sleep for 5 seconds
        //Raise the SIGINT signal
        if (raise(SIGINT)!=0) {
                cerr << "Error raising signal" << endl;
                return 1;
        }
        cout << "This line should not be printed if signal handler exists the program." << endl;
        return 0;
}

rps@rps-virtual-machine:~$ 
```

**SIG_RAISE :-**



```
rps@rps-virtual-machine:~/ayelet$ vim sig_raise.cpp
rps@rps-virtual-machine:~/ayelet$ make sig_raise
g++     sig_raise.cpp   -o sig_raise
rps@rps-virtual-machine:~/ayelet$ ./sig_raise
Raising SIGINTT signal is 5 seconds...
Interrupt signal (2) received.
```



```
#include <iostream>
#include <csignal>
#include <unistd.h>

void signalHandler(int signum) {
        std::cout << "Interrupt signal (" << signum <<") received.\n";
        exit(signum);
}

int main() {
        struct sigaction action;
        action.sa_handler = signalHandler;
        sigemptyset(&action.sa_mask);
        action.sa_flags = 0;

        if (sigaction(SIGINT, &action, nullptr) < 0) {
                std::cerr << "Error registering signal handler" << std::endl;
                return 1;
        }

        std::cout << "Raising SIGINTT signal is 5 seconds..." << std::endl;
        sleep(5);

        if (raise(SIGINT) != 0) {
                std::cerr << "Error raising signal" << std::endl;
                return 1;
        }


        std::cout << "This line should not be printed if signal handler exists the program." << std::endl;
        return 0;
}
```

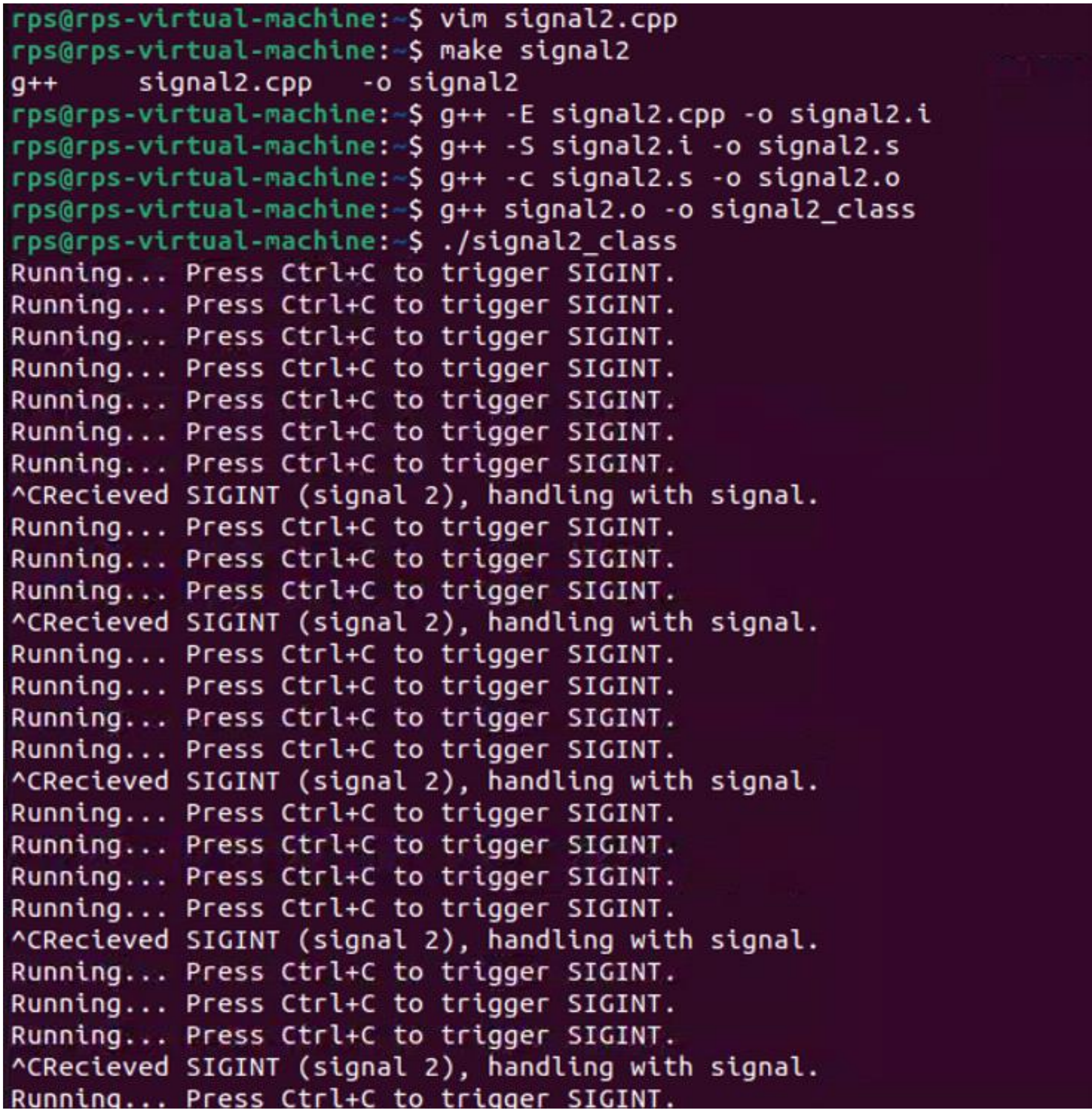
**Basic Handling vs. Advanced Control: Implement signal handling using both signal and sigaction (in separate program runs). Observe the behavior. Which API allows for more control over the signal handler? Explain the key difference in a comment within your code.**

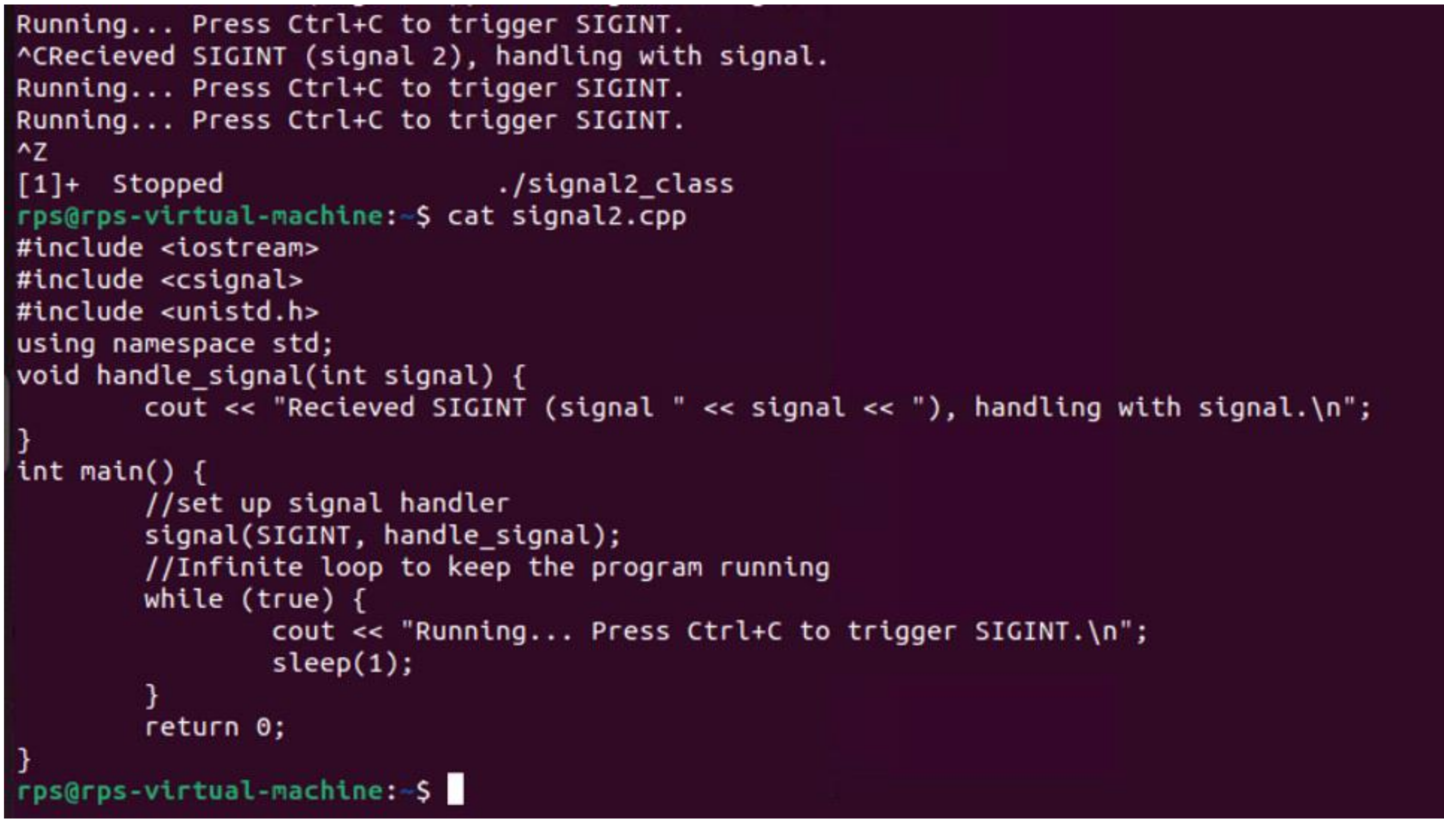
The key difference lies in the flexibility and additional information sigaction() provides compared to signal():

signal() is simpler and easier to use but lacks control over certain aspects of signal handling, especially in complex environments.

sigaction() allows more precise control, such as specifying flags (sa_flags), using a signal handling function with a different prototype (sa_sigaction), and accessing more detailed information about the signal (siginfo_t). This makes sigaction() preferable in scenarios requiring advanced signal handling and information processing.

**Signal:**


```
rps@rps-virtual-machine:~$ vim signal2.cpp
rps@rps-virtual-machine:~$ make signal2
g++     signal2.cpp   -o signal2
rps@rps-virtual-machine:~$ g++ -E signal2.cpp -o signal2.i
rps@rps-virtual-machine:~$ g++ -S signal2.i -o signal2.s
rps@rps-virtual-machine:~$ g++ -c signal2.s -o signal2.o
rps@rps-virtual-machine:~$ g++ signal2.o -o signal2_class
rps@rps-virtual-machine:~$ ./signal2_class
Running... Press Ctrl+C to trigger SIGINT.
Running... Press Ctrl+C to trigger SIGINT.
Running... Press Ctrl+C to trigger SIGINT.
Running... Press Ctrl+C to trigger SIGINT.
Running... Press Ctrl+C to trigger SIGINT.
Running... Press Ctrl+C to trigger SIGINT.
Running... Press Ctrl+C to trigger SIGINT.
^CRecieved SIGINT (signal 2), handling with signal.
Running... Press Ctrl+C to trigger SIGINT.
Running... Press Ctrl+C to trigger SIGINT.
Running... Press Ctrl+C to trigger SIGINT.
^CRecieved SIGINT (signal 2), handling with signal.
Running... Press Ctrl+C to trigger SIGINT.
Running... Press Ctrl+C to trigger SIGINT.
Running... Press Ctrl+C to trigger SIGINT.
Running... Press Ctrl+C to trigger SIGINT.
^CRecieved SIGINT (signal 2), handling with signal.
Running... Press Ctrl+C to trigger SIGINT.
Running... Press Ctrl+C to trigger SIGINT.
Running... Press Ctrl+C to trigger SIGINT.
Running... Press Ctrl+C to trigger SIGINT.
^CRecieved SIGINT (signal 2), handling with signal.
Running... Press Ctrl+C to trigger SIGINT.
Running... Press Ctrl+C to trigger SIGINT.
Running... Press Ctrl+C to trigger SIGINT.
^CRecieved SIGINT (signal 2), handling with signal.
Running... Press Ctrl+C to trigger SIGINT.
```



```
Running... Press Ctrl+C to trigger SIGINT.
^CRecieved SIGINT (signal 2), handling with signal.
Running... Press Ctrl+C to trigger SIGINT.
Running... Press Ctrl+C to trigger SIGINT.
^Z
[1]+  Stopped                 ./signal2_class
rps@rps-virtual-machine:~$ cat signal2.cpp
#include <iostream>
#include <csignal>
#include <unistd.h>
using namespace std;
void handle_signal(int signal) {
        cout << "Recieved SIGINT (signal " << signal << "), handling with signal.\n";
}
int main() {
        //set up signal handler
        signal(SIGINT, handle_signal);
        //Infinite loop to keep the program running
        while (true) {
                cout << "Running... Press Ctrl+C to trigger SIGINT.\n";
                sleep(1);
        }
        return 0;
}
rps@rps-virtual-machine:~$ 
```

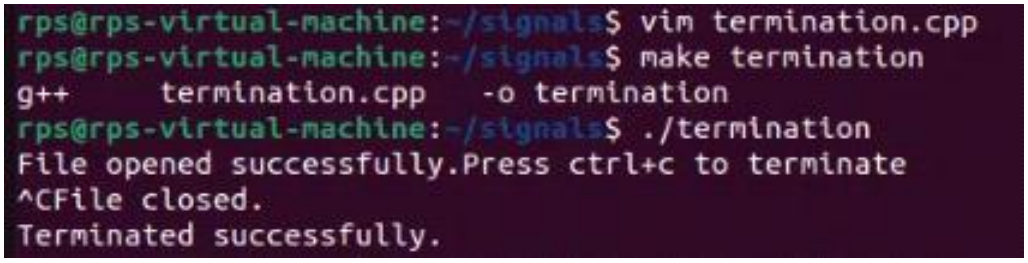
Graceful Termination with Signal Handling

**Objective: Modify your program to demonstrate graceful termination upon receiving a specific signal (e.g., SIGINT). Within the signal handler, perform any necessary cleanup tasks (e.g., closing files, releasing resources) before exiting the program gracefully.**

**Implementation:**

**In your signal handler function, include code to perform cleanup actions. This might involve closing open files, releasing memory, or writing data to disk.**

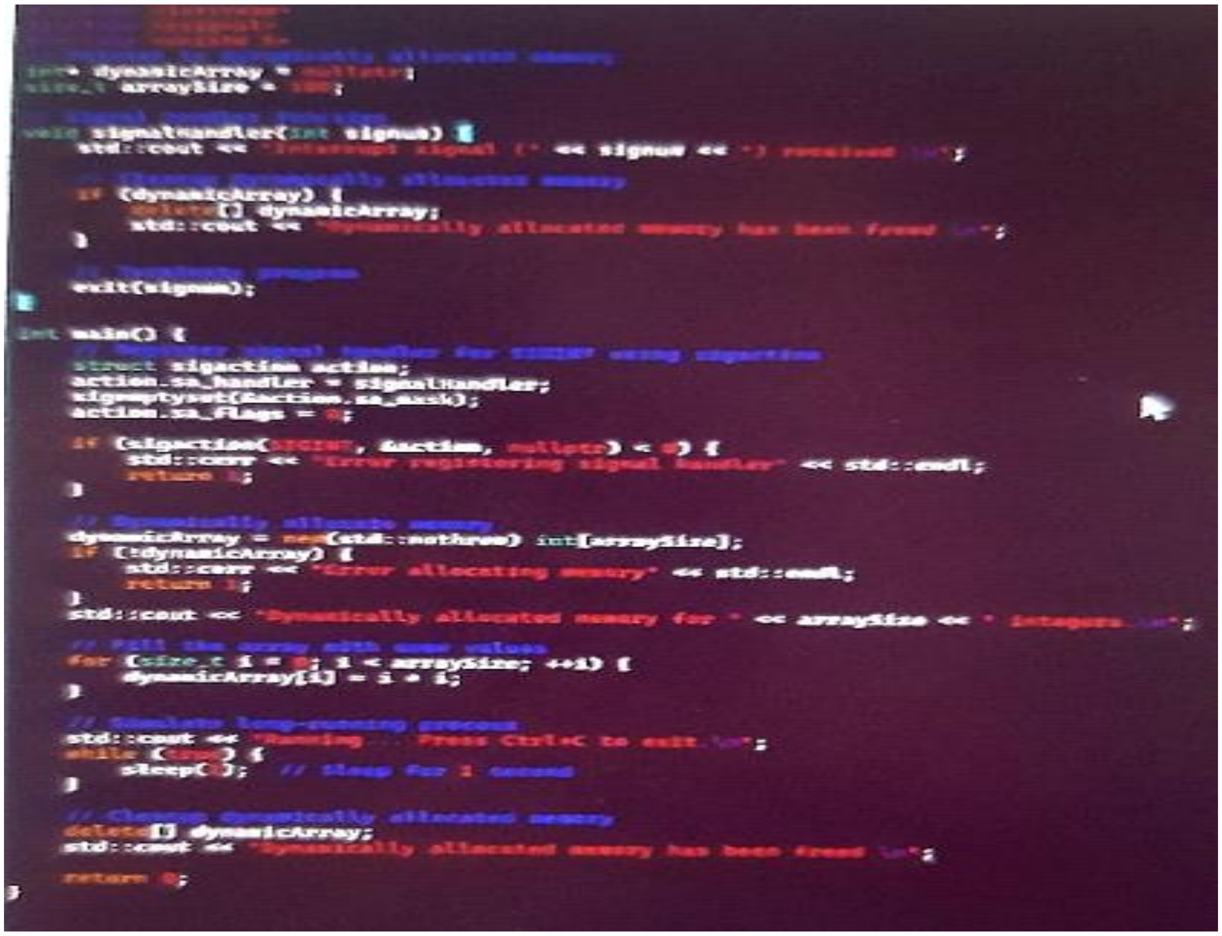
**Use exit(0) or similar methods to terminate the program after cleanup is complete.**



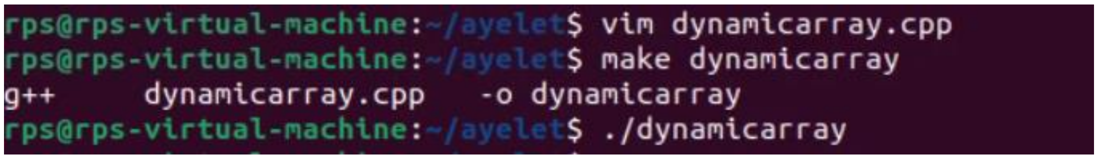
```
rps@rps-virtual-machine:~/signals$ vim termination.cpp
rps@rps-virtual-machine:~/signals$ make termination
g++     termination.cpp    -o termination
rps@rps-virtual-machine:~/signals$ ./termination
File opened successfully.Press ctrl+c to terminate
^CFile closed.
Terminated successfully.
```

```
>Cat termination.cpp

#include <iostream>

#include <fstream>

#include <csignal>

#include <cstdlib>

#include <unistd.h>

using namespace std;

ofstream file;

void signal_handler(int signal) {

    if (signal == SIGINT) {

            if (file.is_open()) {

            file.close();

            cout << "File closed successfully." << endl;        }

        cout << "Program terminated gracefully." << endl;

        exit(0);    }

}

int main() {

    if (signal(SIGINT, signal_handler) == SIG_ERR) {

        cerr << "Error setting up signal handler." << endl;

        return 1;

    }     file.open("example.txt");

    if (!file.is_open()) {

        cerr << "Error opening file" << endl;

        return 1;    }
        cout << "File opened successfully. Press Ctrl+C to terminate." << endl;

        while (true) {

                file << "Writing to file..." << endl;

                file.flush();

                sleep(5);

        }        if (file.is_open()) {

                file.close();

        }       return 0;

}
```

**DYNAMIC MEMORY :-**



```
#include <iostream>
#include <csignal>
#include <unistd.h>
// Pointer to dynamically allocated memory
int* dynamicArray = nullptr;
size_t arraySize = 100;

// Signal handler function
void signalHandler(int signum) {
    std::cout << "Interrupt signal (" << signum << ") received.\n";

    // Cleanup dynamically allocated memory
    if (dynamicArray) {
        delete[] dynamicArray;
        std::cout << "Dynamically allocated memory has been freed.\n";
    }

    // Terminate program
    exit(signum);
}

int main() {
    // Register signal handler for SIGINT using sigaction
    struct sigaction action;
    action.sa_handler = signalHandler;
    sigemptyset(&action.sa_mask);
    action.sa_flags = 0;

    if (sigaction(SIGINT, &action, nullptr) < 0) {
        std::cerr << "Error registering signal handler" << std::endl;
        return 1;
    }

    // Dynamically allocate memory
    dynamicArray = new(std::nothrow) int[arraySize];
    if (!dynamicArray) {
        std::cerr << "Error allocating memory" << std::endl;
        return 1;
    }
    std::cout << "Dynamically allocated memory for " << arraySize << " integers.\n";

    // Fill the array with some values
    for (size_t i = 0; i < arraySize; ++i) {
        dynamicArray[i] = i * i;
    }

    // Simulate long-running process
    std::cout << "Running... Press Ctrl+C to exit.\n";
    while (true) {
        sleep(1);  // Sleep for 1 second
    }

    // Cleanup dynamically allocated memory
    delete[] dynamicArray;
    std::cout << "Dynamically allocated memory has been freed.\n";

    return 0;
}
```



```
rps@rps-virtual-machine:~/ayelet$ vim dynamicarray.cpp
rps@rps-virtual-machine:~/ayelet$ make dynamicarray
g++     dynamicarray.cpp   -o dynamicarray
rps@rps-virtual-machine:~/ayelet$ ./dynamicarray
```