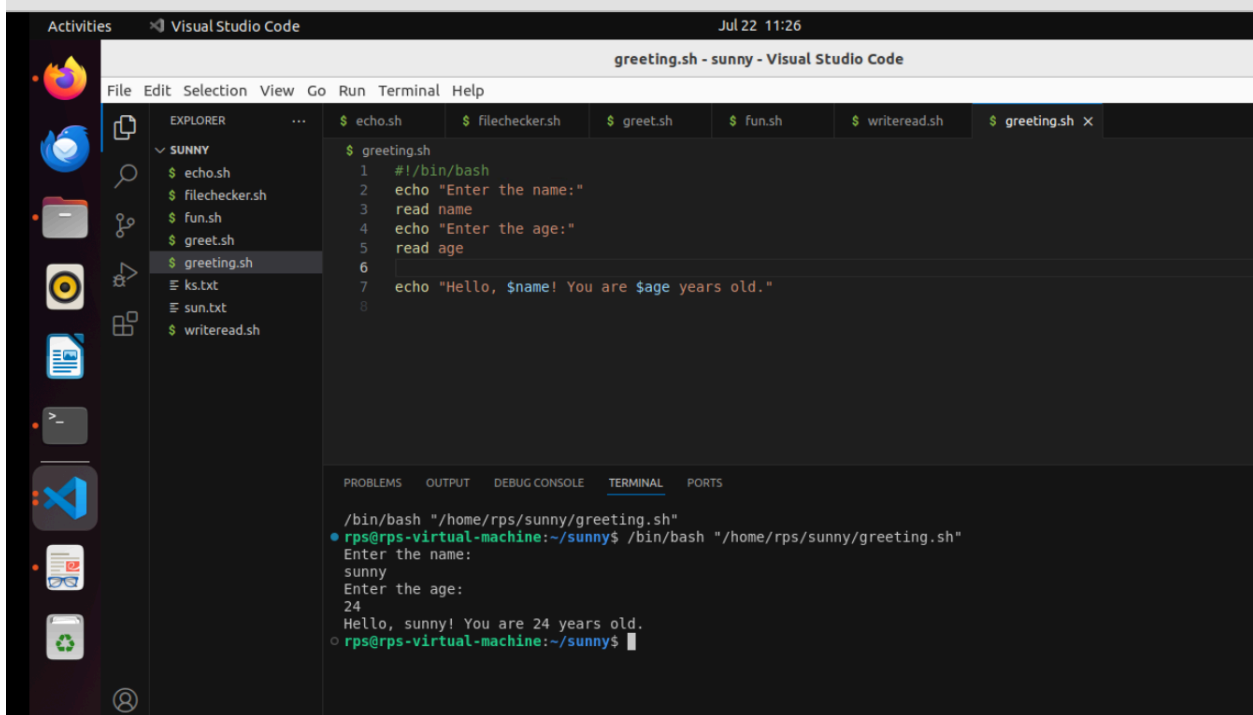


SUNNY KUMAR
22 JULY TASK

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



The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left lists files in a directory named 'SUNNY': `echo.sh`, `filechecker.sh`, `fun.sh`, `greet.sh`, `greeting.sh` (selected), `ks.txt`, `sun.txt`, and `writeread.sh`. The main editor displays the content of `greeting.sh`:

```
1 #!/bin/bash
2 echo "Enter the name:"
3 read name
4 echo "Enter the age:"
5 read age
6
7 echo "Hello, $name! You are $age years old."
8
```

Below the editor, the TERMINAL panel shows the script being executed. The prompt is `rps@rps-virtual-machine:~/sunny$`. The user enters `sunny` for the name and `24` for the age. The script outputs: `Hello, sunny! You are 24 years old.`

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.

The screenshot shows the Visual Studio Code editor with a file named `choice.sh` open. The Explorer sidebar on the left shows a project named `SUNNY` containing several files: `choice.sh`, `echo.sh`, `filechecker.sh`, `fun.sh`, `greet.sh`, `greeting.sh`, `ks.txt`, `sun.txt`, and `writeread.sh`. The `choice.sh` file is selected, and its contents are displayed in the editor. The script is a bash script that takes a choice from the user and performs actions based on that choice. The terminal at the bottom shows the execution of the script, including the prompt to enter a choice and the resulting output.

```
choice.sh - sunny - Visual Studio Code
File Edit Selection View Go Run Terminal Help

EXPLORER
$ choice.sh
$ echo.sh
$ filechecker.sh
$ fun.sh
$ greet.sh
$ greeting.sh
$ choice.sh x

SUNNY
$ choice.sh
1 #!/bin/bash
9
10
11 if [ "$choice" -eq 1 ]; then
12     echo "Listing files."
13     ls
14 elif [ "$choice" -eq 2 ]; then
15     echo "Enter the name of the directory to create: "
16     read DIR_NAME
17     mkdir mydir
18 elif [ "$choice" -eq 3 ]; then
19     echo "Exiting"
20     break
21 else
22     echo "kripiyaaaaaaaa aap sahi wikalp chunel"
23 fi

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

2. Create directory
3. Exit
Enter your choice [1-3]:
1
Listing files.
choice.sh echo.sh filechecker.sh fun.sh greeting.sh greet.sh ks.txt sun.txt writeread.sh
rps@rps-virtual-machine:~/sunny$ /bin/bash "/home/rps/sunny/choice.sh"
Please choose an option:
1. List files
2. Create directory
3. Exit
Enter your choice [1-3]:
5
kripiyaaaaaaaa aap sahi wikalp chunel
rps@rps-virtual-machine:~/sunny$
```

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

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

```
#!/bin/bash
```

SIGNAL HANDLER FUNCTION

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

```
// Signal handler function IN CPP
```

```
void signalHandler(int signum) {
    std::cout << "WHAT EVER Interrupt signal (" << signum << ") received.\n";
    // Cleanup and close up stuff here
    // Terminate program
    exit(signum);
}
```

```

}

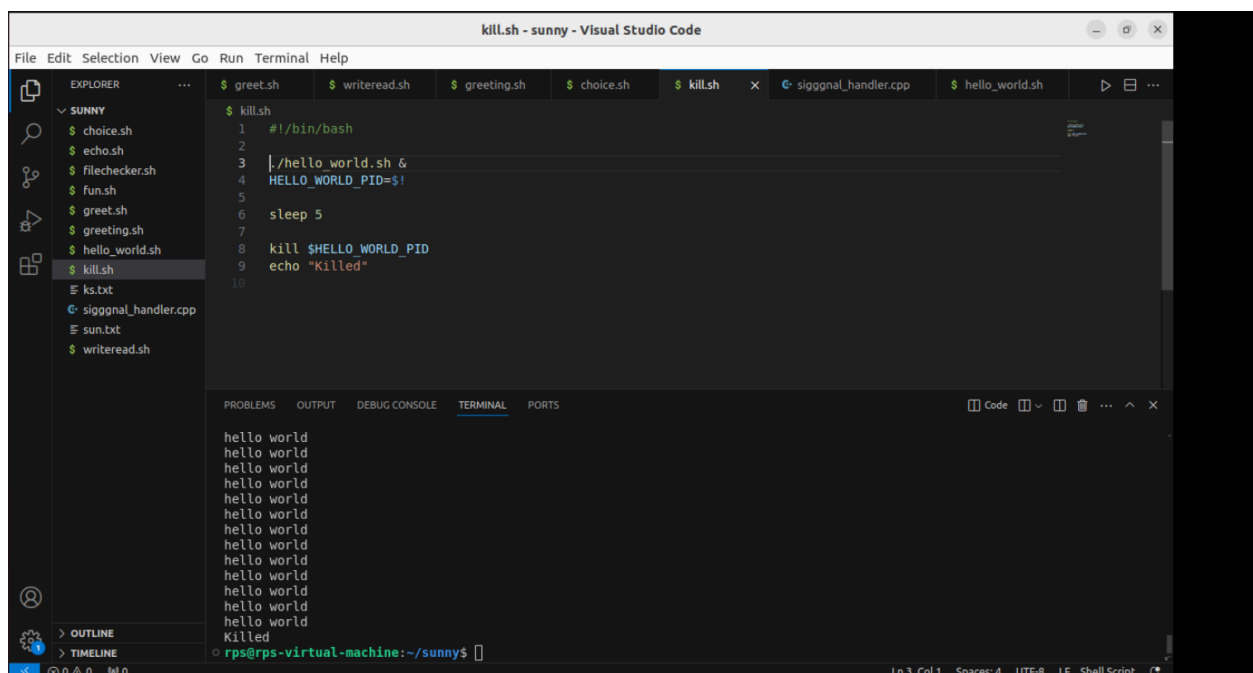
int main() {
    // Register signal handler for SIGINT
    signal(SIGINT, signalHandler);

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

    return 0;
}

```

Q. Write a script that continuously prints "hello world" to the terminal. Additionally, create another script that runs this "hello world" script and terminates it after a specified duration, displaying the message "Killed" upon termination.



The screenshot shows a Visual Studio Code window titled 'kill.sh - sunny - Visual Studio Code'. The Explorer panel on the left shows a file named 'kill.sh' selected. The main editor displays the following script:

```

1  #!/bin/bash
2
3  ./hello_world.sh &
4  HELLO_WORLD_PID=$!
5
6  sleep 5
7
8  kill $HELLO_WORLD_PID
9  echo "Killed"
10

```

The TERMINAL panel at the bottom shows the output of the script:

```

hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
hello world
Killed
rps@rps-virtual-machine:~/sunny$

```

SIGNAL HANDLER FUNCTION

```

#include <iostream>
#include <csignal>

```

```

#include <unistd.h>

// Signal handler function IN CPP

void signalHandler(int signum) {
    std::cout << "WHAT EVER Interrupt signal (" << signum << ") received.\n";
    // Cleanup and close up stuff here
    // Terminate program
    exit(signum);
}

int main() {
    // Register signal handler for SIGINT
    signal(SIGINT, signalHandler);
    signal(SIGSEGV, signalHandler);

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

    return 0;
}

```

SIGACTION

```

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

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

```

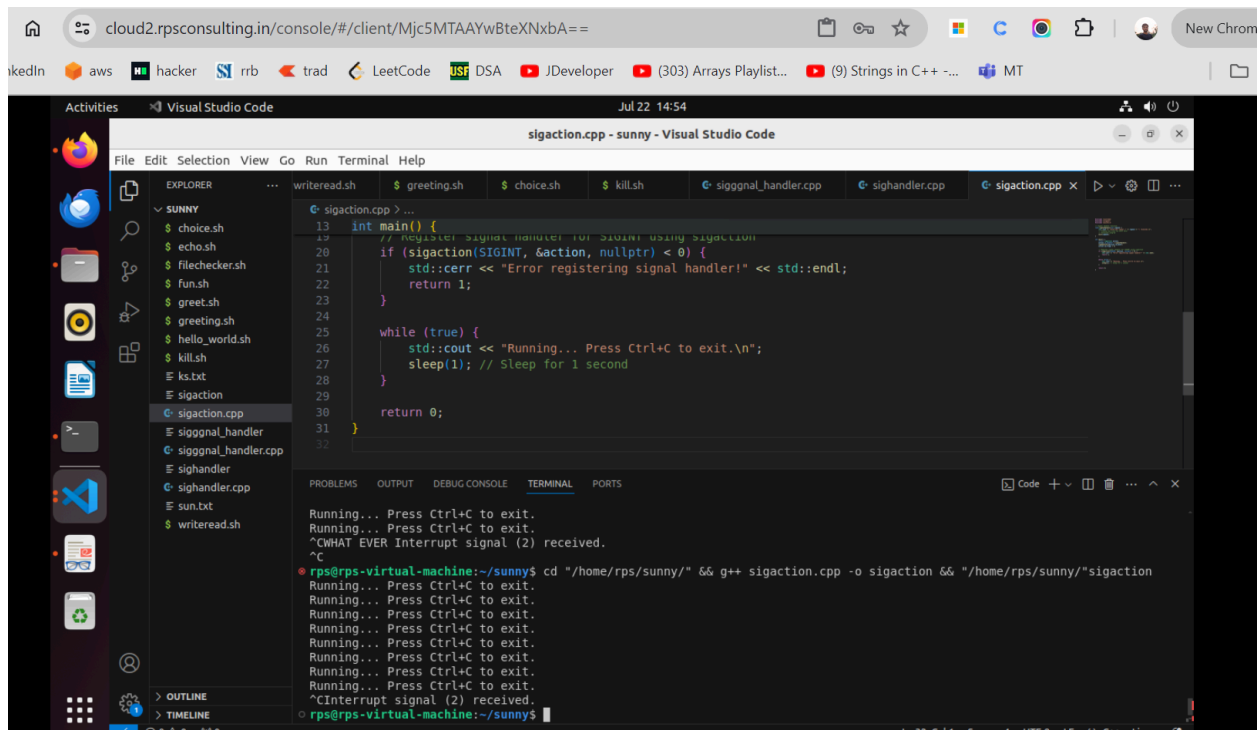
```

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

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

return 0;
}

```



```

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

```

```

// Signal handler function
void signalHandler(int signum) {
    std::cout << "Interrupt signal (" << signum << ") received.\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) {
        std::cerr << "Error registering signal handler" << std::endl;
        return 1;
    }

    std::cout << "Raising SIGINT signal in 5 seconds..." << std::endl;
    sleep(5); // Sleep for 5 seconds

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

    // This part of the code will not be reached if signal handler terminates the program
    std::cout << "This line should not be printed if signal handler exits the program." << std::endl;
    return 0;
}

```

The screenshot shows the Visual Studio Code interface with the file `raise.cpp` open. The code defines a `main` function that prints a message, sleeps for 5 seconds, and then raises the `SIGINT` signal. A signal handler is registered to catch `SIGINT` and print an error message. The terminal shows the execution of `sigaction.cpp` and `raise.cpp`, demonstrating the signal being raised and caught.

```
raise.cpp - sunny - Visual Studio Code
File Edit Selection View Go Run Terminal Help

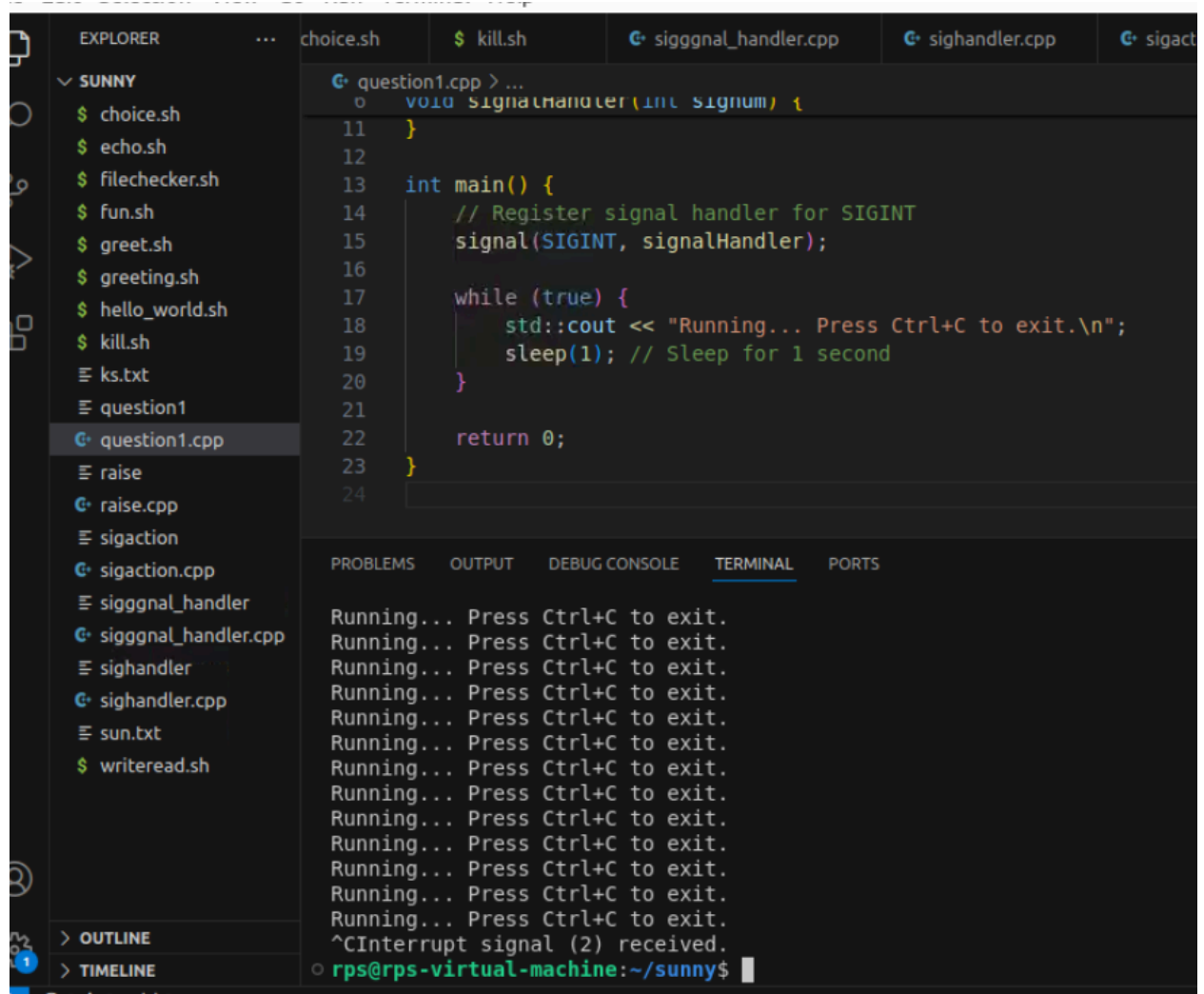
EXPLORER
SUNNY
$ choice.sh
$ echo.sh
$ filechecker.sh
$ fun.sh
$ greet.sh
$ greeting.sh
$ hello_world.sh
$ kill.sh
$ ks.txt
$ raise
$ raise.cpp
$ sigaction
$ sigaction.cpp
$ siggnal_handler
$ siggnal_handler.cpp
$ sighandler
$ sighandler.cpp
$ sun.txt
$ writeread.sh

raise.cpp
13 int main() {
25     std::cout << "Raising SIGINT signal in 5 seconds..." << std::endl;
26     sleep(5); // Sleep for 5 seconds
27
28     // Raise the SIGINT signal
29     if (raise(SIGINT) != 0) {
30         std::cerr << "Error raising signal" << std::endl;
31         return 1;
32     }
33
34     // This part of the code will not be reached if signal handler terminates the program
35     std::cout << "This line should not be printed if signal handler exits the program." << std::endl;
36     return 0;
37 }
38

TERMINAL
^C
rps@rps-virtual-machine:~/sunny$ cd "/home/rps/sunny/" && g++ sigaction.cpp -o sigaction && "/home/rps/sunny/sigaction"
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.
^CInterrupt signal (2) received.
rps@rps-virtual-machine:~/sunny$ cd "/home/rps/sunny/" && g++ raise.cpp -o raise && "/home/rps/sunny/raise"
Raising SIGINT signal in 5 seconds...
Interrupt signal (2) received.
rps@rps-virtual-machine:~/sunny$
```

Q.

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.



EXPLORER

SUNNY

\$ choice.sh

\$ echo.sh

\$ filechecker.sh

\$ fun.sh

\$ greet.sh

\$ greeting.sh

\$ hello_world.sh

\$ kill.sh

\$ ks.txt

\$ ques2

ques2.cpp

question1

question1.cpp

raise

raise.cpp

sigaction

sigaction.cpp

sigggna_handler

sigggna_handler.cpp

sighandler

sighandler.cpp

sun.txt

writeread.sh

OUTLINE

TIMELINE

kill.sh

sigggna_handler.cpp

sighandler.cpp

sigaction.cpp

raise.cpp

question1.cpp

ques2.cpp > ...

```
13 int main() {
14     // Register signal handler for SIGINT using sigaction
15     if (sigaction(SIGINT, &action, nullptr) < 0) {
16         std::cerr << "Error registering signal handler!" << std::endl;
17         return 1;
18     }
19     while (true) {
20         std::cout << "Running... Press Ctrl+C to exit.\n";
21         sleep(1); // Sleep for 1 second
22     }
23     return 0;
24 }
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

```
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.
^CInterrupt signal (2) received.
rps@rps-virtual-machine:~/sunny$
```