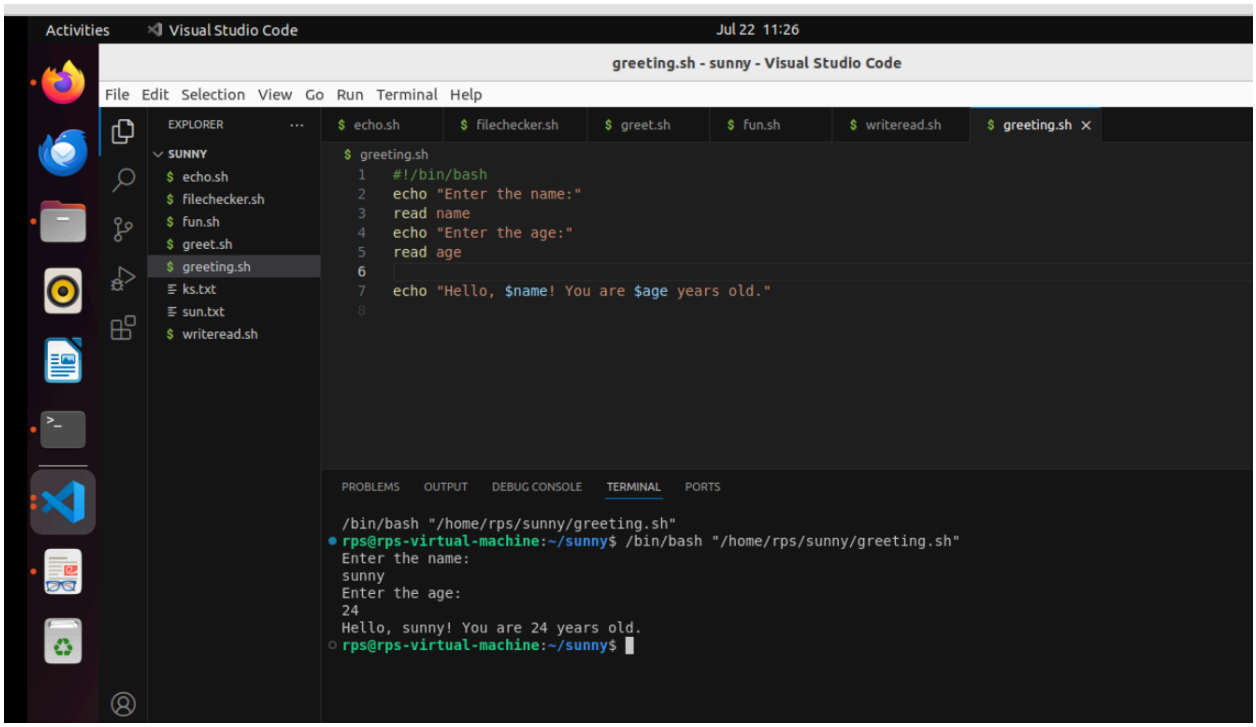
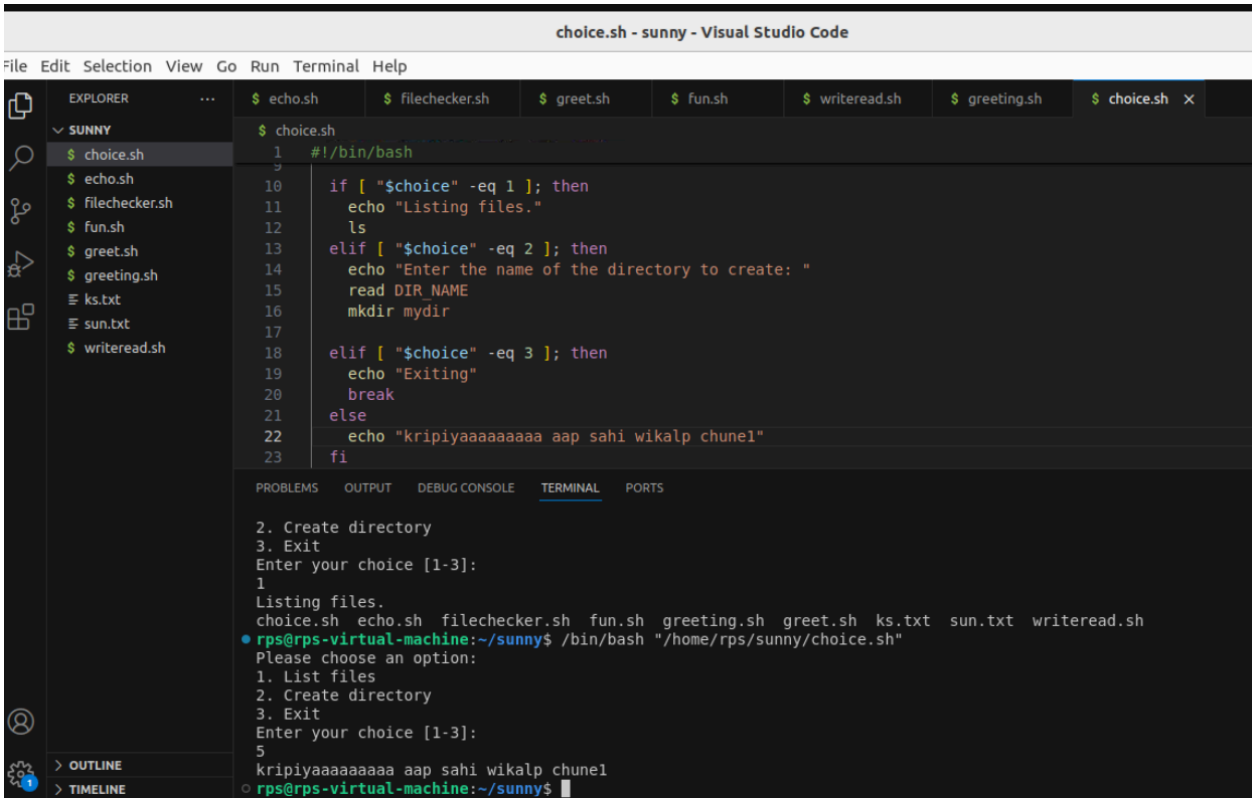
SUNNY KUMAR
22 JULY TASK

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



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.



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

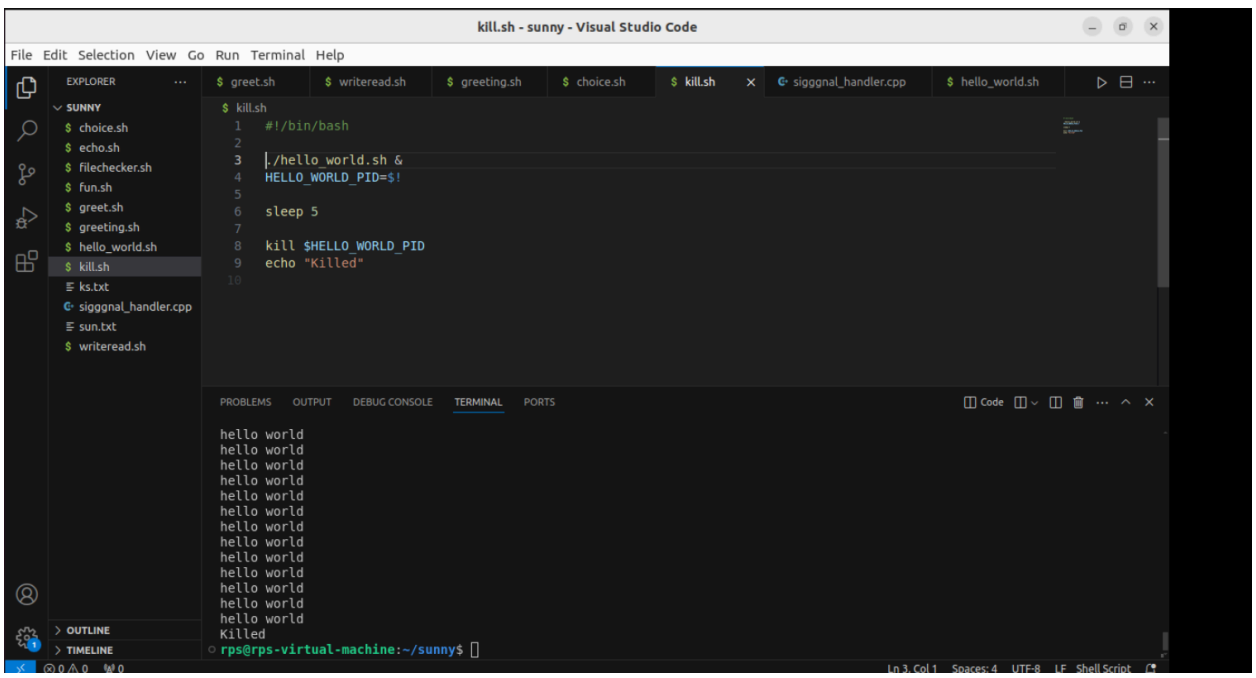
Create a script that takes a text file as input and replaces all occurrences of a specific word with another word.
#!/bin/bash

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.



SIGNAL HANDLER FUNCTION

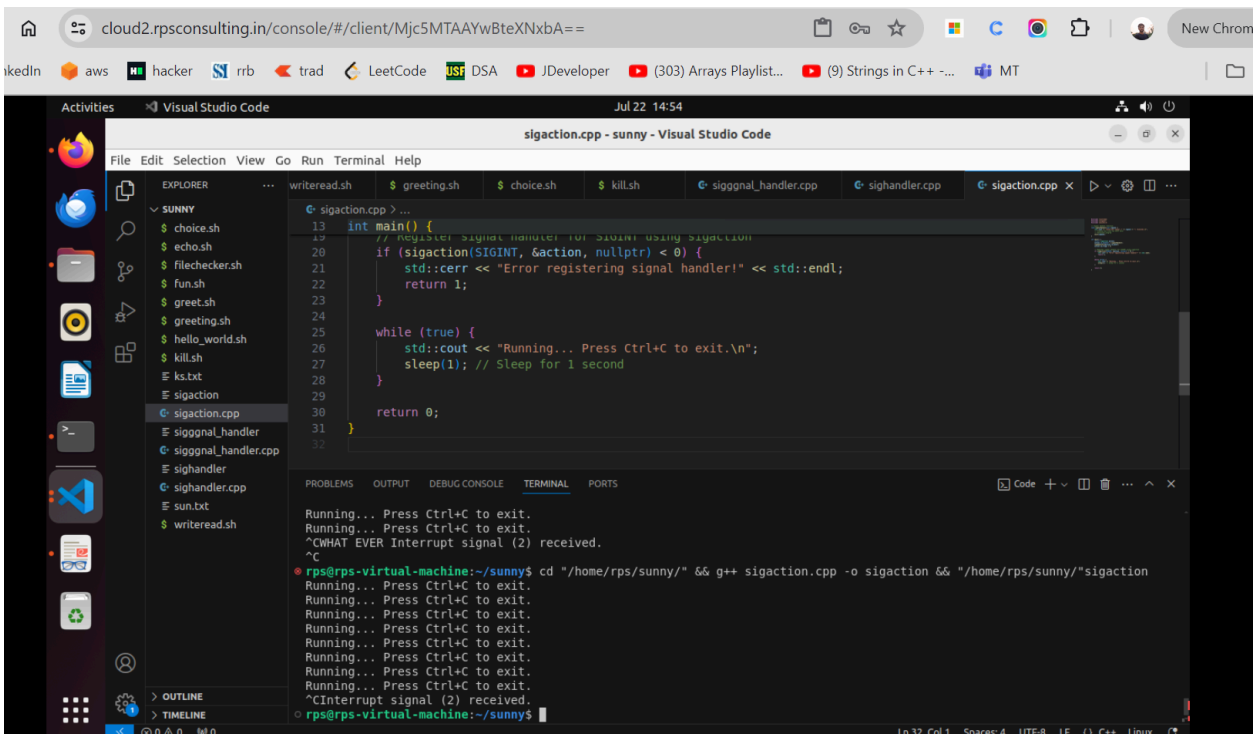
```
#include <iostream>
#include <csignal>
```

```
#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

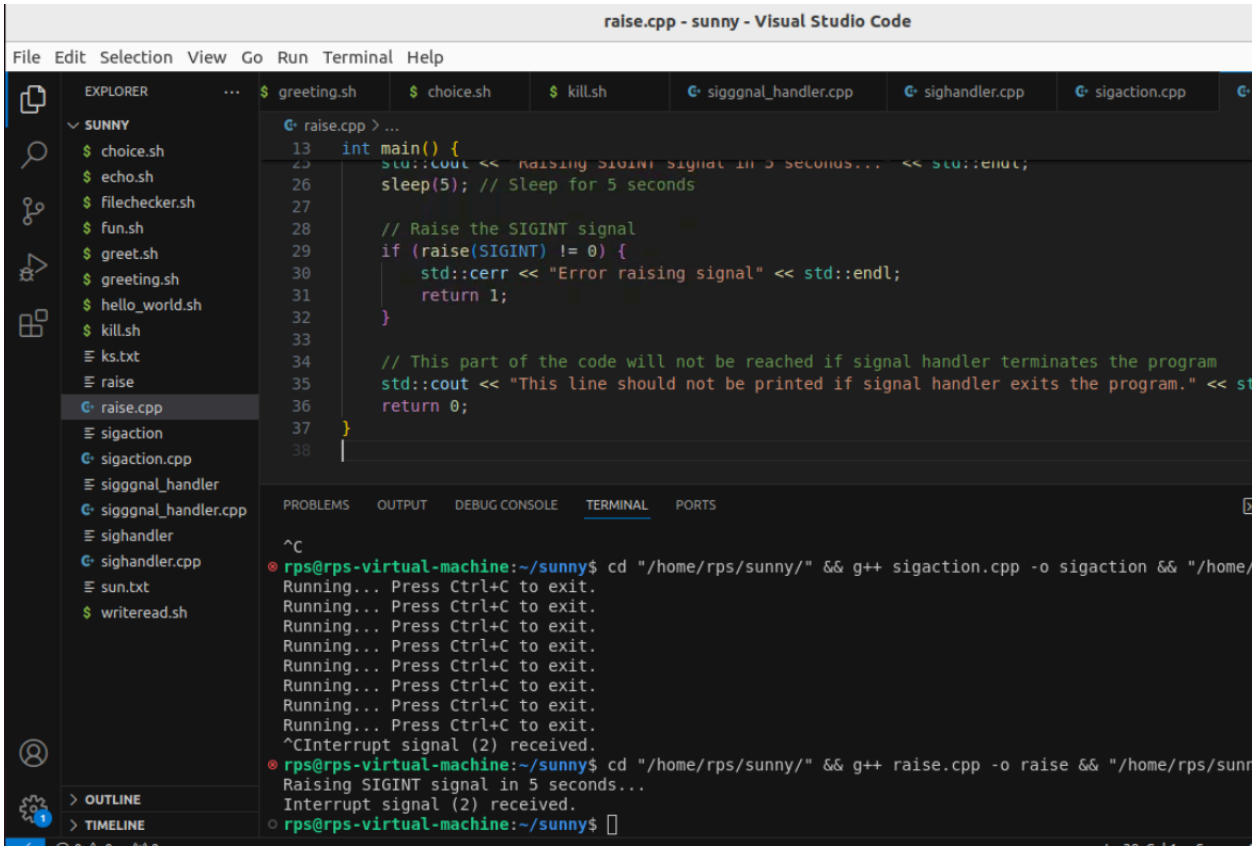
```
#include <iostream>
#include <csignal>
#include <unistd.h>

// 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;
}
```



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
- question1
- question1.cpp
- raise
- raise.cpp
- sigaction
- sigaction.cpp
- sigggnal_handler
- sigggnal_handler.cpp
- sighandler
- sighandler.cpp
- sun.txt
- writeread.sh

OUTLINE

TIMELINE

choice.sh | kill.sh | sigggnal_handler.cpp | sighandler.cpp | sigact

question1.cpp > ...

```cpp
void signalHandler(int 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;
}
```

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

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
- sigggnal_handler
- sigggnal_handler.cpp
- sighandler
- sighandler.cpp
- sun.txt
- writeread.sh

OUTLINE

TIMELINE

kill.sh | sigggnal_handler.cpp | sighandler.cpp | sigaction.cpp | raise.cpp | questio

ques2.cpp > ...

```cpp
int main() {

    // 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;
}
```

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