

Q.Simple Signal Handler: Write a C++ program that handles the SIGINT signal (Ctrl+C) gracefully by printing a custom message before exiting.

The screenshot displays the Visual Studio Code interface with the following components:

- File Explorer (Left Panel):** Shows a project named "SUNNY". The file list includes:
  - choice.sh
  - echo.sh
  - filechecker.sh
  - fun.sh
  - greet.sh
  - greeting.sh
  - hello\_world.sh
  - interrupted
  - interrupted.cpp
  - kill.sh
  - ks.txt
  - ques2
  - ques2.cpp
  - question1
  - question1.cpp
  - raise
  - raise.cpp
  - sigaction
  - sigaction.cpp
  - sigggnal\_handler
  - sigggnal\_handler.cpp
  - sighandler
  - sighandler.cpp
  - simple
  - simple.cpp
- Code Editor (Main Area):** Displays the source code for `simple.cpp`.

```
1 // simple.cpp > ...  
2 void signalHandler(int signum) {  
3     // ...  
4 }  
5  
6 int main() {  
7     // Register signal handler for SIGINT  
8     signal(SIGINT, signalHandler);  
9  
10    while (true) {  
11        std::cout << "Running... Press Ctrl+C to exit.\n";  
12        sleep(1); // Sleep for 1 second  
13    }  
14  
15    return 0;  
16 }
```
- Terminal (Bottom Panel):** Shows the output of running the program.

```
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.  
^CSIGINT (Ctrl+C) signal received. Exiting gracefully...  
rps@rps-virtual-machine:~/sunny$
```

Q. Multiple Signal Handling: Create a program that handles both SIGINT and SIGTERM signals, printing a different message for each.

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

```
// Signal handler function for multiple signals
void signalHandler(int signum) {
    if (signum == SIGINT) {
        std::cout << "SIGINT (Ctrl+C) signal received. Exiting gracefully...\n";
    } else if (signum == SIGTERM) {
        std::cout << "SIGTERM signal received. Exiting gracefully...\n";
    }
    exit(signum);
}
```

```

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

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

    return 0;
}

```

The screenshot shows the Visual Studio Code interface with the file `multiple.cpp` open. The code in the editor is as follows:

```

1 //Multiple Signal Handling: Create a program that handles both SIGINT and SIGTERM signals, print
2
3 #include <iostream>
4 #include <csignal>
5 #include <unistd.h>
6
7 // Signal handler function for multiple signals
8 void signalHandler(int signum) {
9     if (signum == SIGINT) {
10         std::cout << "SIGINT (Ctrl+C) signal received. Exiting gracefully...\n";
11     } else if (signum == SIGTERM) {
12         std::cout << "SIGTERM signal received. Exiting gracefully...\n";
13     }
14     exit(signum);
15 }
16

```

The terminal output shows the program running and printing the prompt "Running... Press Ctrl+C to exit or send SIGTERM to terminate." multiple times. Finally, it receives a SIGINT signal (Ctrl+C) and prints "SIGINT (Ctrl+C) signal received. Exiting gracefully..." before exiting.

Q. Ignoring Signals: Develop a program that ignores the SIGTERM signal and continues execution even after it's sent.

```

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

```

```
// Signal handler function for SIGINT
void signalHandler(int signum) {
    std::cout << "SIGINT (Ctrl+C) signal received. Exiting gracefully...\n";
    exit(signum);
}

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

    // Ignore SIGTERM signal
    signal(SIGTERM, SIG_IGN);

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

    return 0;
}
```

The screenshot shows a C++ IDE with a file explorer on the left and a terminal at the bottom. The file explorer lists several shell scripts and C++ files. The terminal shows the execution of a program that ignores SIGTERM signals. The program's output indicates that it is running and can be terminated by pressing Ctrl+C or sending a SIGTERM signal, which it ignores. The program then prints a message indicating that it has received a SIGINT signal and is exiting gracefully.

```
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
$ ignore
$ ignore.cpp
$ interrupted
$ interrupted.cpp
$ kill.sh
$ ks.txt
$ multiple
$ multiple.cpp
$ ques2
$ ques2.cpp
$ question1
$ question1.cpp
$ raise
$ raise.cpp
$ sigaction
$ sigaction.cpp
$ sigsignal_handler
$ sigsignal_handler.cpp

IGNORE.cpp > ...
1
2
3 // ignoredQ. Ignoring Signals: Develop a program that ignores the SIGTERM sig
4
5 #include <iostream>
6 #include <csignal>
7 #include <unistd.h>
8
9 // Signal handler function for SIGINT
10 void signalHandler(int signum) {
11     std::cout << "SIGINT (Ctrl+C) signal received. Exiting gracefully...\n";
12     exit(signum);
13 }
14
15 int main() {
16     // Register signal handler for SIGINT
17
18     iRunning... Press Ctrl+C to exit or send SIGTERM (ignored) to terminate.
19     gRunning... Press Ctrl+C to exit or send SIGTERM (ignored) to terminate.
20     Running... Press Ctrl+C to exit or send SIGTERM (ignored) to terminate.
21     Running... Press Ctrl+C to exit or send SIGTERM (ignored) to terminate.
22     Running... Press Ctrl+C to exit or send SIGTERM (ignored) to terminate.
23     Running... Press Ctrl+C to exit or send SIGTERM (ignored) to terminate.
24     Running... Press Ctrl+C to exit or send SIGTERM (ignored) to terminate.
25     Running... Press Ctrl+C to exit or send SIGTERM (ignored) to terminate.
26     Running... Press Ctrl+C to exit or send SIGTERM (ignored) to terminate.
27     Running... Press Ctrl+C to exit or send SIGTERM (ignored) to terminate.
28     Running... Press Ctrl+C to exit or send SIGTERM (ignored) to terminate.
29     Running... Press Ctrl+C to exit or send SIGTERM (ignored) to terminate.
30     Running... Press Ctrl+C to exit or send SIGTERM (ignored) to terminate.
31     ^CSIGINT (Ctrl+C) signal received. Exiting gracefully...
32     rps@rps-virtual-machine:~/sunny$
```

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

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

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

    // Define the signal set to block
    sigset_t sigSet;
    sigemptyset(&sigSet);
    sigaddset(&sigSet, SIGTERM);
```

```

// Block SIGTERM
if (sigprocmask(SIG_BLOCK, &sigSet, nullptr) != 0) {
    std::cerr << "Failed to block SIGTERM.\n";
    return 1;
}

std::cout << "SIGTERM is blocked during critical section.\n";

// Critical section starts
std::cout << "Entering critical section...\n";
std::cout << "Simulating critical section...\n";
sleep(5); // Simulating critical section work
std::cout << "Exiting critical section...\n";
// Critical section ends

// Unblock SIGTERM
if (sigprocmask(SIG_UNBLOCK, &sigSet, nullptr) != 0) {
    std::cerr << "Failed to unblock SIGTERM.\n";
    return 1;
}

std::cout << "SIGTERM is unblocked.\n";

// Sleep to demonstrate signal handling after unblocking
sleep(10);

return 0;
}

```

```
block.cpp > ...
1  #include <iostream>
2  #include <csignal>
3  #include <unistd.h>
4
5  // Signal handler function for SIGTERM
6  void signalHandler(int signum) {
7      std::cout << "Signal (" << signum << ") received.\n";
8  }
9
10 int main() {
11     // Register signal handler for SIGTERM
12     signal(SIGTERM, signalHandler);
13
14     // Define the signal set to block
15     sigset_t sigSet;
16     sigemptyset(&sigSet);
17
18     // Block SIGTERM
19     sigprocmask(SIG_BLOCK, &sigSet, NULL);
20
21     // Simulate critical section
22     while(1) {
23         // ...
24     }
25
26     // Unblock SIGTERM
27     sigprocmask(SIG_UNBLOCK, &sigSet, NULL);
28
29     // ...
30 }

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
cd "/home/rps/sunny/" && g++ block.cpp -o block && "/home/rps/sunny/"block
rps@rps-virtual-machine:~/sunny$ cd "/home/rps/sunny/" && g++ block.cpp -o block && "/home/rps/sunny/"block
SIGTERM is blocked during critical section.
Entering critical section...
Simulating critical section...
Exiting critical section...
SIGTERM is unblocked.
rps@rps-virtual-machine:~/sunny$
```

```
#include <iostream>
#include <csignal>
#include <cstdio>
#include <unistd.h>
#include <thread>
#include <chrono>
```

```
// File name to clean up
const char *fileName = "tempfile.txt";
```

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

    // Cleanup and close up stuff here
    if (std::remove(fileName) == 0) {
        std::cout << "File " << fileName << " deleted successfully.\n";
    } else {
        std::perror("Error deleting file");
    }

    // Terminate program
    exit(signum);
}
```

```

int main() {
    // Create a file to be removed
    FILE *file = std::fopen(fileName, "w");
    if (file == nullptr) {
        std::perror("Error creating file");
        return 1;
    }

    std::fputs("Temporary file content.", file);
    std::fclose(file);
    std::cout << "File " << fileName << " created.\n";

    // Register signal handler for SIGINT
    std::signal(SIGINT, signalHandler);
    std::signal(SIGTERM, signalHandler);

    std::cout << "Press Ctrl+C to trigger the signal handler...\n";

    // Infinite loop to keep the program running
    while (true) {
        // Sleep to prevent high CPU usage
        std::this_thread::sleep_for(std::chrono::seconds(1));
    }

    return 0;
}

```

