

QUESTION 10 ANSWERS

The screenshot shows a terminal window titled "Terminal - coder@9acced2abea8: ~". The window contains a C program with syntax highlighting. The code includes headers for stdio.h, stdlib.h, unistd.h, signal.h, and sys/wait.h. It defines two signal handlers: handle_sigterm and handle_sigint. The main function creates two children using fork(). If the first child fails, it exits. If successful, it sleeps for 5 seconds, prints its PID, sends a SIGTERM to its parent, and then exits. The second child follows a similar process but sleeps for 10 seconds. Both children then loop, printing their PID and sleeping for 1 second. The terminal window has a standard Linux-style header with File, Edit, View, Terminal, Tabs, and Help menus.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <signal.h>
#include <sys/wait.h>

void handle_sigterm(int sig) {
    printf("\nParent received SIGTERM. Exiting gracefully.\n");
    exit(0);
}

void handle_sigint(int sig) {
    printf("\nParent received SIGINT. Exiting gracefully.\n");
    exit(0);
}

int main() {
    pid_t child1, child2;
    printf("Parent PID: %d\n", getpid());
    signal(SIGTERM, handle_sigterm);
    signal(SIGINT, handle_sigint);

    child1 = fork();
    if (child1 < 0) {
        perror("fork failed");
        exit(1);
    } else if (child1 == 0) {
        sleep(5);
        printf("Child 1 sending SIGTERM to parent (PID %d)\n", getppid());
        kill(getppid(), SIGTERM);
        exit(0);
    }

    child2 = fork();
    if (child2 < 0) {
        perror("fork failed");
        exit(1);
    } else if (child2 == 0) {
        sleep(10);
        printf("Child 2 sending SIGTERM to parent (PID %d)\n", getppid());
        kill(getppid(), SIGTERM);
        exit(0);
    }

    while (1) {
        printf("Parent running...PID:%d\n",getpid());
        sleep(1);
    }
    return 0;
}
```

I implemented a C program where the parent process runs indefinitely and installs handlers for `SIGTERM` and `SIGINT`. Two child processes send `SIGTERM` after 5 seconds and `SIGINT` after 10 seconds using `kill()`. The parent handles each signal differently and exits gracefully. The output screen shot is not included as it gets killed each time bash is run.