Quiz#2
AI-5A

Time: **<u>5+1 mins</u>**

Q1# Discuss the use of the ***`kill()`*** system call compared to the ***`kill`*** command. Write a code snippet where a parent process sends a ***`SIGTERM`*** signal to a child process using ***`kill()`*** and explain how each parameter in ***`kill(int, int)`*** is determined and used. What would happen if the parent waits after sending the signal?

<u>Sol:</u>

The `kill()` system call in C provides similar functionality to the `kill` command, allowing signals to be sent programmatically. This function takes two parameters:

- **First Parameter:** The signal type, specified as an integer (e.g., `SIGTERM` or its integer equivalent, 15).
- **Second Parameter:** The process ID (PID) of the target process.

Below is a solution where a parent process creates a child process using `fork()`, sends a `SIGTERM` signal to the child using `kill()`, and then waits for the child to terminate.

Example Code:

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

int main() {
    pid_t pid = fork();

    if (pid < 0) {
        perror("Fork failed");
        exit(1);
    } else if (pid == 0) {
        // Child process
        printf("Child process running with PID: %d\n", getpid());
        while (1) {}  // Infinite loop to keep child alive for demonstration
    } else {
        // Parent process
        sleep(2);  // Give child time to initialize
        printf("Parent sending SIGTERM to child\n");
        kill(pid, SIGTERM);  // Parent sends SIGTERM to child
        wait(NULL);  // Parent waits for child process to finish
        printf("Child process terminated\n");
    }

    return 0;
}
```

Explanation:

1. **Creating the Child Process:** The `fork()` call creates a child process. The child enters an infinite loop, remaining active.
2. **Sending SIGTERM:** After a short sleep, the parent sends `SIGTERM` to the child using `kill(pid, SIGTERM);`.
3. **Waiting for the Child to Terminate:** The `wait(NULL);` call in the parent ensures it waits until the child process completes after receiving `SIGTERM`.

Outcome:

When executed, the child process will print its PID and continue running. After 2 seconds, the parent sends `SIGTERM`, which terminates the child. The parent then outputs "Child process terminated" after the child exits.