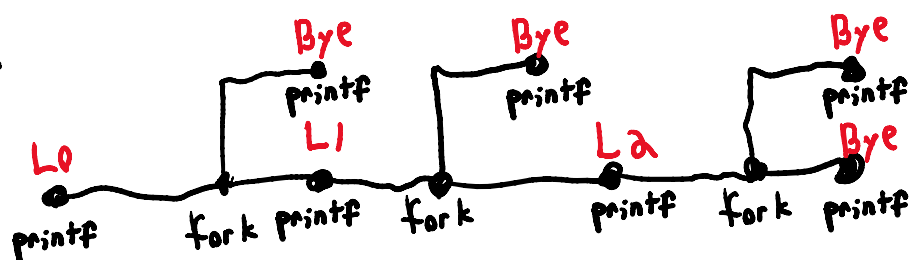


Tuesday, December 6, 2022 9:44 AM

1.



The diagram illustrates the execution of a recursive function `f` that prints "Bye" and calls `fork` and `printf`. The execution flow is shown as a sequence of steps across three levels of recursion: `L0`, `L1`, and `L2`.

Level `L0`:

- Initial state: `printf`
- Step 1: `fork` (creates a new process)
- Step 2: `printf` (prints "Bye")

Level `L1`:

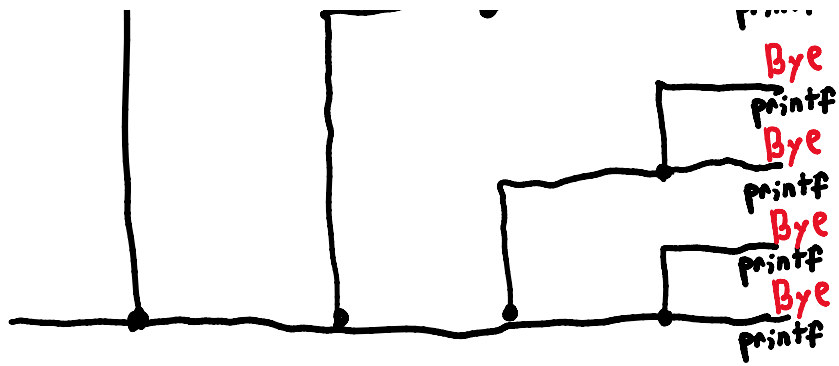
- Step 1: `printf` (prints "Bye")
- Step 2: `fork` (creates a new process)
- Step 3: `printf` (prints "Bye")

Level `L2`:

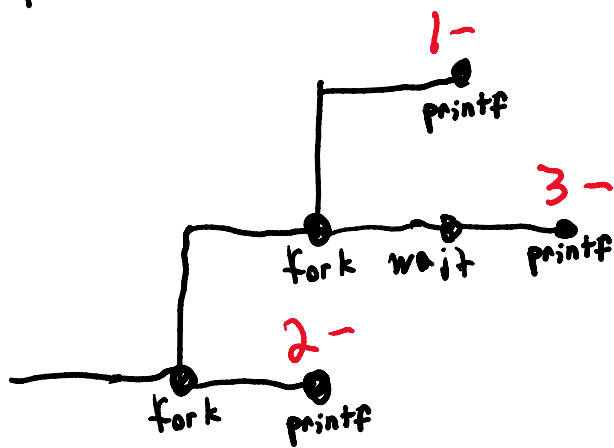
- Step 1: `printf` (prints "Bye")
- Step 2: `fork` (creates a new process)
- Step 3: `printf` (prints "Bye")

The diagram shows the flow of execution from `L0` to `L1` to `L2`, and then back to `L0` and `L1` as the recursive calls return.

The diagram illustrates a recursive function call stack for a function named 'Bye'. The vertical line on the left represents the call stack. Horizontal lines branch off to the right, representing function calls. Each branch contains a box with 'printf' and 'Bye' written inside. The boxes are arranged in a staircase pattern, showing the sequence of calls and returns. The text 'Bye' is written in red, and 'printf' is written in black.



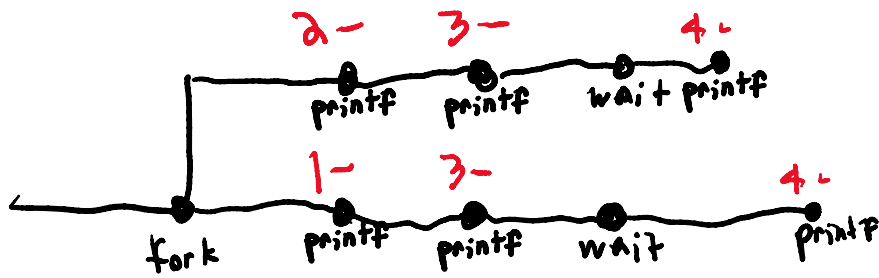
6,



Possible outputs:

- 1-
- 3-
- 2-
- 2-
- 1-
- 3-

7,



Possible outputs:

1-

3-

2-

3-

4-

4-

2-

3-

4-

1-

3-

4-

Reason:

when we fork, we don't know whether the parent or child process will be run first, and thus we don't know which order the print statements will occur in