Dylan Phoutthavong

April 24th, 2025

CSCI 3751

Homework Assignment #4 (40 points)

1. (15 points) Read the section 6.10 Time and Date routines. Write a C (or C++) program called timeLeft implementing the following:
    a. display the current date/time in the format specified below in yellow (Fig 6.9 and 6.10)
    b. ask the user to enter a future target date in "mm/dd/yyyy" format (Figure 6.12)
    c. display remaining days, hours, minutes, and seconds to reach the user specified target date from the current date and time
    d. display the total number of seconds to reach the user specified target date from the current date/time

```
[phouttdy@csci-gnode-02 Homework4]$ ./timeLeft
Current date and time: 2025-04-26 Saturday 09:25:21 PM, MDT
Please enter a target date in the future(mm/dd/yyyy): 12/25/2025
Until the target date: 243 Days, 0 Hours, 0 Minutes, 0 Seconds
Total number of seconds of the target date: 20995200 seconds
[phouttdy@csci-gnode-02 Homework4]$ ./timeLeft
Current date and time: 2025-04-26 Saturday 09:25:27 PM, MDT
Please enter a target date in the future(mm/dd/yyyy): 04/26/2026
Until the target date: 365 Days, 0 Hours, 0 Minutes, 0 Seconds
Total number of seconds of the target date: 31536000 seconds
[phouttdy@csci-gnode-02 Homework4]$
```

(Extra credit: 5 points) Verify that the total number of seconds from the question 4 above actually converts to the answer of question 3. You may write a small C/C++ program or MS Excel formulas to verify it.

```
[phouttdy@csci-gnode-02 Homework4]$ ./timeLeftExtra
Current date and time: 2025-04-26 Saturday 08:49:48 PM, MDT
Please enter a target date in the future(mm/dd/yyyy): 12/25/2025
Until the target date: 243 Days, 0 Hours, 0 Minutes, 0 Seconds
Total number of seconds of the target date: 20995200 seconds
Verification Passed :) : Breakdown matches total seconds.
[phouttdy@csci-gnode-02 Homework4]$ ./timeLeftExtra
Current date and time: 2025-04-26 Saturday 08:50:07 PM, MDT
Please enter a target date in the future(mm/dd/yyyy): 04/26/2026
Until the target date: 365 Days, 0 Hours, 0 Minutes, 0 Seconds
Total number of seconds of the target date: 31536000 seconds
Verification Passed :) : Breakdown matches total seconds.
[phouttdy@csci-gnode-02 Homework4]$
```

2. (5 points) Read the problem 7.10 on page 226 and answer the following questions. (Hint: The answer is “No”.)
    a. Explain in detail why it’s not correct.

    The code attempts to return a value that was an output from a pointer to a local automatic variable inside a conditional block. Specifically:

    ```
    if (val == 0) {
      int val;
      val = 5;
      ptr = &val;
    }
    return(*ptr + 1);
    ```

    The variable val declared inside the if block is automatic and local to that block. Once the block ends, val goes out of scope and its memory becomes invalid. Therefore, the pointer ptr now points to memory that is no longer safe or predictable to use. Accessing or dereferencing it causes undefined behavior.

    b. Give your solution and describe why your solution would work.

One way to fix this is to define the variable outside the block so its lifetime extends until the function ends:

```
int f1(int val) {
  int num = 0;
  int *ptr = &num;

  if (val == 0) {
    static int safe_val;
    safe_val = 5;
    ptr = &safe_val;
  }
  return (*ptr + 1);
```

```
}
```

We can also use heap allocation:

```
int f1(int val) {
   int *ptr = malloc(sizeof(int));
   *ptr = (val == 0) ? 5 : 0;
   int result = *ptr + 1;
   free(ptr);
   return result;
}
```

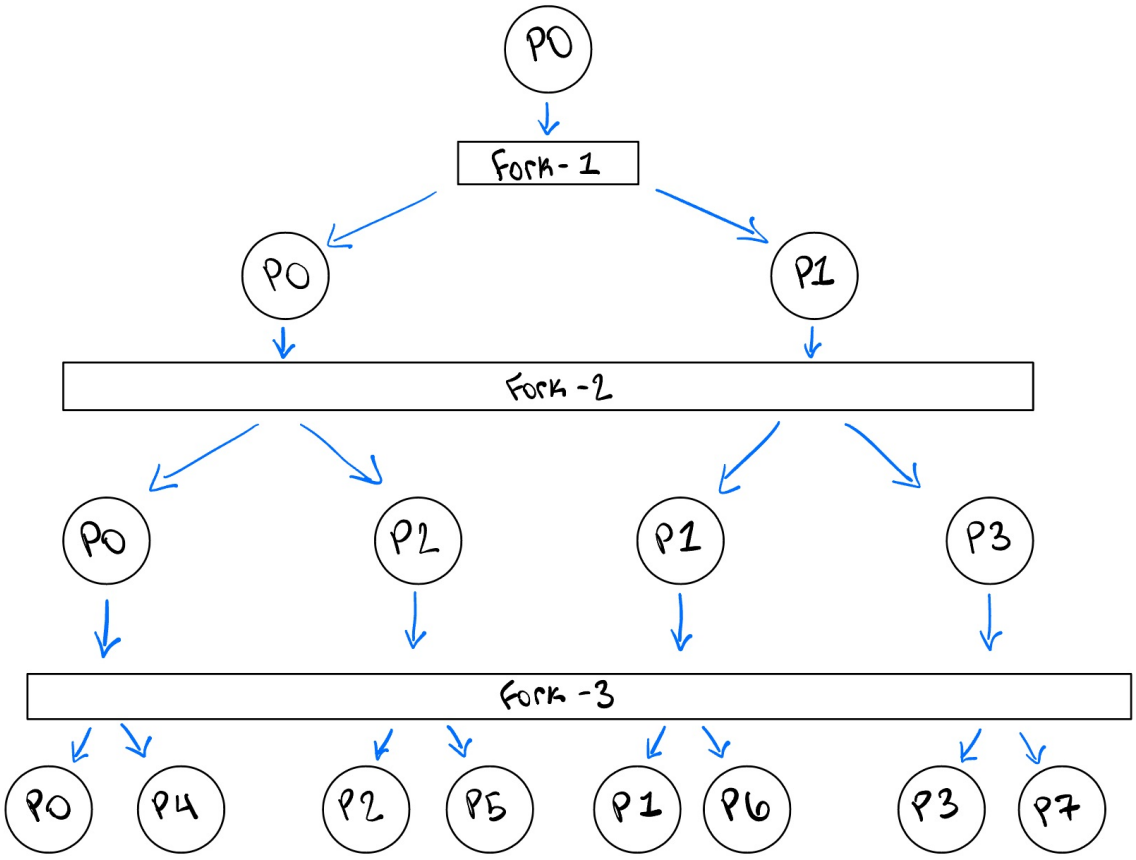
Both makes sure that ptr references valid memory when dereferenced.

3. (10 points) Read the section 8.3 “fork Function”.
    a. Summarize the important points of the section. (5 points)
        i. fork() is a system call to create a new child process from an existing parent process.
        ii. It is called once but returns twice - once in the parent and once in the child.
        iii. In the child process, fork() returns 0; in the parent, it returns the child's PID.
        iv. Child and parent processes continue executing independently after fork().
        v. Modern UNIX systems use Copy-on-Write (COW) to optimize memory use after a fork().
        vi. File descriptors, environment variables, and process memory are copied, but some attributes like process ID (PID) differ.
        vii. Differences include return values, process IDs, and reset of timing counters.
        viii. Parent and child can share file descriptors initially but operate independently after forking unless explicitly managed.

    b. After a program executes the following series of fork() calls, how many new processes will result (assuming that none of the calls fails)? Draw a high-

level simple diagram depicting the new process creation sequence. (5 points)

```
fork();
fork();
fork();
```



4. (10 points) Read the section 8.13 “system Function” section in its entirety.
    a. Describe what this function is and summarize how it works. (5 points) Feel free to use any references you can find and to draw diagrams if helpful to answer the question.
        i. system() runs a shell command from a C/C++ program.
        ii. Internally uses fork(), exec(), and waitpid().
        iii. Format: int system(const char *cmdstring);
        iv. Returns the command’s termination status.

v. When invoked, system() creates a child process. The child executes the shell and passes the command to it. The parent waits for it to finish.
vi. It is convenient but less efficient than calling exec() directly.

b. Search the Internet and list the advantages and disadvantages of system() calls. Also, write your own guidelines on when system() calls may and may not be used? (5 points)
    i. Advantages:
        1. Easy to use for shell commands
        2. Useful for prototyping or scripting tasks
        3. Automatically handles fork/exec/wait
    ii. Disadvantages:
        1. Security risk (especially with user input in command string)
        2. Slower than exec()
        3. Can't customize process attributes like exec() can
        4. Behavior depends on the system shell
    iii. Guidelines: Use system() when:
        1. Command is static and known
        2. Simplicity is more important than performance or security
    iv. Avoid system() when:
        1. Accepting user input as part of command string
        2. Performance and control over process are needed
        3. Portability and shell independence are important