CS29206 Systems Programming Laboratory, Spring 2022–2023

Programming Assignment 5

Date: 06–April–2023 (to be submitted in the lab)

Topic: Shell programming using bash

In this assignment, you write two **executable** bash scripts. The first script involves simple integer arithmetic. One usually does not write shell scripts for this type of problems, but this script is meant for reviewing your familiarity with the bash syntax. The second script is a typical application that a system administrator would write in order to monitor user activities on a Linux server. You too can spy on other users with this script.

Part 1: *Zeckendorf representation of positive integers*

The Zeckendorf representation of a positive integer N is

$$N = F(t_1) + F(t_2) + \cdots + F(t_k),$$

where $F(t)$ is the t-th Fibonacci number, and $t_1 > t_2 > \cdots > t_k \geqslant 1$. One additionally has $t_i \geqslant t_{i+2} + 2$ for all i, in this representation. Write a bash script **`fibrep.sh`** that computes and prints the Zeckendorf representation of a positive integer N supplied by the user. Your script should proceed as follows.

The script first reads N from the first command-line argument. If the user runs the script without a command-line argument or with a command-line argument which is not a positive integer, the script should exit after issuing an error message. If the user supplies multiple command-line arguments, only the first one is used.

The script computes the Fibonacci sequence $F(0), F(1), F(2), \ldots, F(t)$ until it locates $F(t) \geqslant N$. The script then keeps on trying to decrement N by $F(t), F(t-1), F(t-2), \ldots$ (in that sequence). A subtraction of $F(i)$ from N is allowed if and only if the current value of N is $\geqslant F(i)$. The loop stops as soon as N reduces to 0. All indices found during this phase are reported as shown in the sample output below. After the printing, the script exits with the success status.

Sample output

```
$ ./fibrep.sh
Run fibrep.sh with a single positive integer argument
$ ./fibrep.sh -123
Invalid argument -123 to fibrep.sh
$ ./fibrep.sh 55
Computed Fibonacci numbers up to F(10) = 55
55 = F(10)
$ ./fibrep.sh 555
Computed Fibonacci numbers up to F(15) = 610
555 = F(14) + F(12) + F(9)
$ ./fibrep.sh 1234567890987654321
Computed Fibonacci numbers up to F(89) = 1779979416004714189
1234567890987654321 = F(88) + F(83) + F(80) + F(78) + F(75) + F(73) + F(62) +
F(60) + F(57) + F(55) + F(51) + F(49) + F(46) + F(43) + F(38) + F(36) + F(34)
+ F(32) + F(29) + F(25) + F(20) + F(18) + F(11) + F(8)
$
```

Part 2: *Login statistics on a server*

Write a bash script **`loginfo.sh`** that displays the list of users that logged on to the server along with their respective login counts. Users that did not log on to the server should not be reported. You are urged to experiment on the department's compute servers (using your account) because many users use those machines, whereas the desktop machine or your personal computer (laptop) has very few users.

Before writing the code, study the Unix command **`last`** and its command-line options. This command prints a chronological listing of the (successful) login instances made by the users in a limited period of time. Read the man page of **`last`** to know its command-line options. Run the command with various options. When you are done, proceed as detailed below.

Your script **`loginfo.sh`** may be run with an optional command-line argument. Without any command-line argument, the entire last-log output is processed. If there is a command-line argument *N* (a positive integer), then **`last`** should be called with the directive to print only *N* last login lines.

Your script first invokes **`last`** with the relevant command-line argument(s) (as the case demands), and stores the output of **`last`** in a string *S*. The **`last`** output contains multiple lines, so *S* is a string containing new-line characters. Your script should split the string *S* into its lines, and store these lines in an array *A*. Write a loop to get the lines from *S* one by one using the bash command **`read`** along with the redirection **`<<<`**. After the loop, print how many login lines have been read. The last two lines of a **`last`** output do not store login information. So the number of login lines is the size of the array *A* minus 2.

Extract the username from each line stored in *A* using a regular-expression-based search. Store the login counts of different users, in an associative array *H* indexed by the usernames. Do not use **`grep`** or **`wc`** or any external file in your script.

Finally, print the list of users that logged on to the server and the login counts accumulated in the associative array *H*. There is ***no*** need to print the list in any particular order (like sorted with respect to usernames or login counts). Use the format as given in the sample output below. This pertains to the server **`10.5.18.70`** (usernames are modified for privacy). Notice that each IITKGP roll number contains ***nine*** letters. If you see only 8-letter roll numbers in your output, it is a problem with your script. Repair it.

Sample output

```
$ ./loginfo.sh
855 login records read
21FB10161 logged in 1 times
18FB30153 logged in 1 times
21FB10188 logged in 1 times
21FB10564 logged in 3 times
21FB10195 logged in 8 times
21FB10607 logged in 3 times
21FB30912 logged in 1 times
21FB10345 logged in 6 times
...
reboot logged in 3 times
...
21FB10299 logged in 9 times
...
abhij logged in 4 times
...
root logged in 10 times
20FB14461 logged in 3 times
...
$ ./loginfo.sh 20
20 login records read
18FB30075 logged in 1 times
20FB10256 logged in 4 times
20FB10808 logged in 1 times
22FB60R98 logged in 1 times
18FB60509 logged in 4 times
abhij logged in 3 times
19FB30099 logged in 4 times
21FB10567 logged in 1 times
20FB51P54 logged in 1 times
$
```

Submit the two individual scripts **`fibrep.sh`** and **`loginfo.sh`** written by you.
