

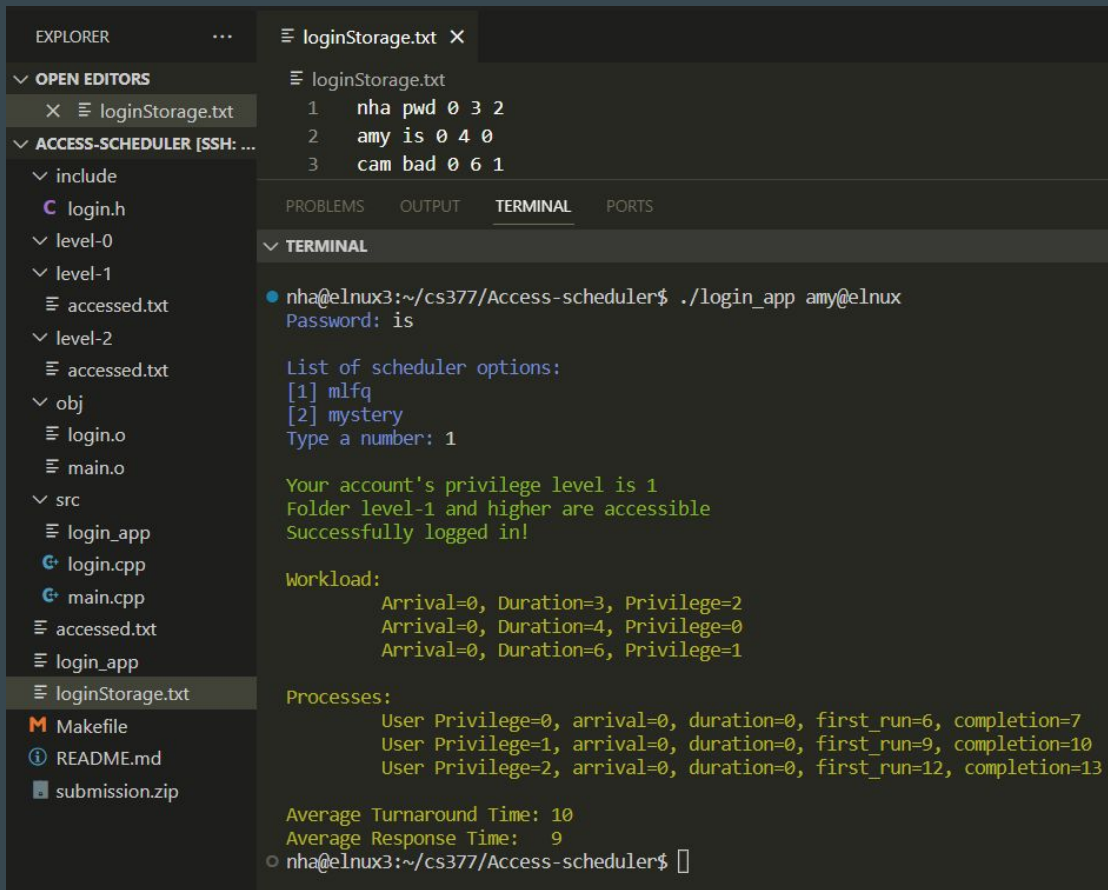
Access Scheduler

...

Nhi Ha
CS 377
Final Project

What is this project about?

- Extension of schedulers with a MLFQ and a mystery scheduler
- Integration of schedulers with user login authentication and their privilege levels
- Traverses directories based on user's privilege of each most efficient processes



The screenshot displays the Visual Studio Code interface. On the left, the Explorer sidebar shows a project structure with folders like 'include', 'level-0', 'level-1', 'level-2', 'obj', 'src', and files like 'login.h', 'accessed.txt', 'login.o', 'main.o', 'login_app', 'login.cpp', 'main.cpp', 'accessed.txt', 'login_app', 'loginStorage.txt', 'Makefile', 'README.md', and 'submission.zip'. The center editor shows the content of 'loginStorage.txt' with three lines of text: 'nha pwd 0 3 2', 'amy is 0 4 0', and 'cam bad 0 6 1'. On the right, the TERMINAL panel shows the output of running './login_app amy@elinux'. The output includes a password prompt, a list of scheduler options (mlfq and mystery), a confirmation of privilege level 1, and a workload summary table.

```
loginStorage.txt X
loginStorage.txt
1  nha pwd 0 3 2
2  amy is 0 4 0
3  cam bad 0 6 1

PROBLEMS OUTPUT TERMINAL PORTS
▼ TERMINAL
● nha@elinux3:~/cs377/Access-scheduler$ ./login_app amy@elinux
Password: is

List of scheduler options:
[1] mlfq
[2] mystery
Type a number: 1

Your account's privilege level is 1
Folder level-1 and higher are accessible
Successfully logged in!

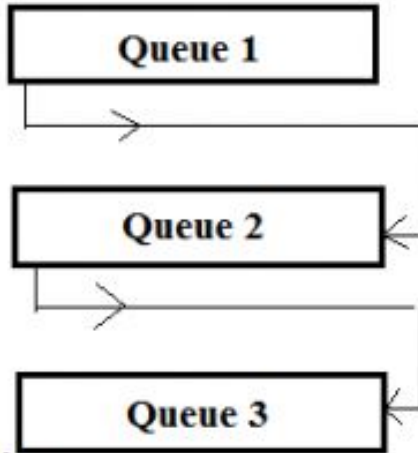
Workload:
Arrival=0, Duration=3, Privilege=2
Arrival=0, Duration=4, Privilege=0
Arrival=0, Duration=6, Privilege=1

Processes:
User Privilege=0, arrival=0, duration=0, first_run=6, completion=7
User Privilege=1, arrival=0, duration=0, first_run=9, completion=10
User Privilege=2, arrival=0, duration=0, first_run=12, completion=13

Average Turnaround Time: 10
Average Response Time: 9
○ nha@elinux3:~/cs377/Access-scheduler$
```

MLFQ Explanation

High Priority



Low Priority

- **Rule 1:** If $\text{Priority}(A) > \text{Priority}(B)$, A runs (B doesn't).
- **Rule 2:** If $\text{Priority}(A) = \text{Priority}(B)$, A & B run in round-robin fashion using the time slice (quantum length) of the given queue.
- **Rule 3:** When a job enters the system, it is placed at the highest priority (the topmost queue).
- **Rule 4:** Once a job uses up its time allotment at a given level (regardless of how many times it has given up the CPU), its priority is reduced (i.e., it moves down one queue).
- **Rule 5:** After some time period S , move all the jobs in the system to the topmost queue.

Sources:

<https://www.geeksforgeeks.org/multilevel-feedback-queue-scheduling-mlfq-cpu-scheduling/>

<https://pages.cs.wisc.edu/~remzi/OSTEP/cpu-sched-mlfq.pdf>

Privilege Levels Accessibility and Workload

```
✓ level-0
✓ level-1
  ≡ accessed.txt
✓ level-2
  ≡ accessed.txt
```

- The lower the number -> The higher the privilege level of the folder.
- The txt means that the folder has already been accessed by the user based on privilege level

```
loginStorage.txt X
loginStorage.txt
1  nha  pwd  0  3  0
2  amy  is  0  4  2
3  cam  bad  0  6  1
4
```

Column 1 -> Username

Column 2 -> Password

Column 3 -> Start time

Column 4 -> Duration

Column 6 -> Privilege levels (holder)

Why integrate privilege level and schedulers

- Can have process priorities and privilege levels go hand-to-hand
- assign priority levels to processes with any user privilege number, then adjust priorities based on the processes behavior using the scheduling algorithm
- A better user interaction to schedulers using login authentication and user's privilege levels