Network Programming Homework 0

Deadline: Wednesday, 2021/10/06 23:55

I. Description

In this homework, you are given a part of a program and a terminal output of two processes executed simultaneously. You should figure out possible context switch points according to the output.

The left part of page 2 contains a program that periodically reads a number from a file, prints its PID with the number, increases the number by one, and writes the number back to the file. The right part of page 2 shows the output of two processes executed simultaneously (suppose there is only one core). Please analyze the possible situation of every context switch point (marked as (1), (2) ... (9)) and give your answer on page 3.

The **Executing Block** means the section where the process is paused when a context switch happens. You should answer with **"Line x ~ y"** which represents "line x is finished and line y must not have been executed". For example, "Line 6 ~ 7" means that "line 6 is finished while line 7 must not have been executed", and "Line 10 ~ 6" means that "line 10 is finished while line 6 (next iteration) must not have been executed". The **Description** block should contain the reason of your judgement.

You should give the answer based on the output in page 2. DO NOT execute the program. Besides, there may be more than one answer, and you only need to give one reasonable answer with explanation.

II. Submission

- You should submit your answer to the E3 system.
- Only submit the answer page.
- You should name your file as **[student_id].pdf**, for example, "0856000.pdf".
  ATTENEION! We only accept PDF format.
- Late submissions are not accepted after the deadline.
- DO NOT use handwriting.

Code

```cpp
int main() {
    fstream seq_file("seqno.txt");
    int seqno, pid = getpid();
    for (int i = 0; i < 20; ++i) {
        seq_file.seekg(ios::beg); /* rewind before read */
        seq_file >> seqno;
        cout << "pid=" << pid << ", seq#=" << seqno << endl;
        ++seqno;
        seq_file.seekp(ios::beg); /* rewind before write */
        seq_file << seqno << endl;
    }
    seq_file.close();
}
```

Output

```
pid=186, seq#=1          pid=186, seq#=13
pid=186, seq#=2          pid=186, seq#=14
(1)                      pid=186, seq#=15
pid=187, seq#=3          pid=186, seq#=16
pid=187, seq#=4          pid=186, seq#=17
pid=187, seq#=5          (7)
(2)                      pid=187, seq#=18
pid=186, seq#=6          pid=187, seq#=19
pid=186, seq#=7          pid=187, seq#=20
pid=186, seq#=8          pid=187, seq#=21
(3)                      (8)
pid=187, seq#=9          pid=186, seq#=18
pid=187, seq#=10         pid=186, seq#=19
pid=187, seq#=11         pid=186, seq#=20
(4)                      pid=186, seq#=21
pid=186, seq#=9          pid=186, seq#=22
pid=186, seq#=10         pid=186, seq#=23
pid=186, seq#=11         (9)
pid=186, seq#=12         pid=187, seq#=22
(5)                      pid=187, seq#=23
pid=187, seq#=12         pid=187, seq#=24
pid=187, seq#=13         pid=187, seq#=25
pid=187, seq#=14         pid=187, seq#=26
pid=187, seq#=15         pid=187, seq#=27
(6)
```

Network Programming Homework 0

Answer

Student ID: 310605004

	Executing Blocks		**Description**
	`pid = 186`	`pid = 187`	
(1)	Line 10 ~ 6		由於(1)之後 pid 187 輸出為 3，可知已將 seqno 寫入檔案。 由於(2)之後 pid 186 輸出為 6，可知此時尚未讀取檔案。
(2)		Line 10 ~ 6	由於(2)之後 pid 186 輸出為 6，可知已將 seqno 寫入檔案。 由於(3)之後 pid 187 輸出為 9，可知此時尚未讀取檔案。
(3)	Line 6 ~ 7		由於(3)之後 pid 187 輸出為 9，可知已將 seqno 寫入檔案。由於(4)之後 pid 186 輸出為 9，並且從(4)看出 10 之後會被寫入檔案，不可能再讀出 9，所以可以確認已經在(3)讀取檔案。
(4)		Line 6 ~ 7	(6)之後可以看到 pid 186 的 13 必來自(5)之前的讀取，這表示(5)之前 pid 186 已經將 13 寫入檔案，也表示(5)之後 pid 187 的 12 是來自於(4)之前讀取的檔案。但這時候有兩種可能，一種是 pid 187 在 4 之前就完成檔案讀取(6~7) 或是在回到(5)的時候才將 12 寫入檔案並讀取(7~10)
(5)	Line 6 ~ 7		由於(6)之後 pid 186 輸出為 13，所以可以確認這個 13 來自(5) 之前的讀取。 (因為(6)之前可以確認檔案至少被寫到 15，不可能有 13)
(6)		Line 7 ~ 6	由於(6)之後 pid 186 的 13 來自(5)之前的讀取，所以是否寫入檔案並不影響。 但由於(7)之後 pid 186 輸出 18，所以可以確定(6)之前 pid 187 沒有讀取檔案。
(7)	Line 6 ~ 7		由於(7)之後 pid 187 輸出為 18，可知已將 seqno 寫入檔案。 由於(8)之後 pid 186 輸出為 18，可知已經讀取檔案。
(8)		Line6 ~ 7	由於(9)之後 pid 187 輸出為 22，可知已經讀取檔案。
(9)	Line 7 ~ end		最後一次輸出結束。