OS 2019 Homework 3

B10632026 四資工三甲 吳苡瑄

1. Please prove the correctness of the following extended compare_and_wait program in terms of mutual exclusion, progress, and bounded waiting.

```
while (true) {
   waiting[i] = true;
   key = 1;
   while (waiting[i] && key == 1)
      key = compare_and_swap(&lock,0,1);
   waiting[i] = false;

    // critical section
   j = (i + 1) % n;
   while ((j != i) && !waiting[j])
      j = (j + 1) % n;
   if (j == i)
      lock = 0;
   else
      waiting[j] = false;
   // remainder section
}
```

Ans.

mutual exclusion:

要進到 critical section 有兩個可能，一個是 **key!=1**，一個是 **waiting[i] = false**，**key** 只有在 **lock==0** 的時候才可能會發生**!=1**，而只有一個 process 發現了 **lock==0** 以後進到 critical section 並會利用 **compare_and_swap** 把 **lock** 重新變回**!=0**，所以只會同時有一個 process 在 critical section，然後 **waiting[i] = false** 是在有 process 離開 citical section 的時候才會發生，所以同時只會有一個，符合 mutual exculsion。

progress:

在一個 process 離開 critical section 的時候只會做 **lock = 0** 和 **waiting[j] = false** 其中一個而已，而這兩者任一發生時都可以讓其他 process 進到 critical section，所以符合 progress。

bounded waiting:

每次 process 離開 critical section 時都是往下面一個 id 的 process 去找符合條件的，只要符合就讓他進去 critical section，而他們一直一個往下找一個，所有 process 最多都不會等超過 **n-1** 次，所以符合。

2. Regarding Bakery algorithm, please answer following questions

```
do {
    Choosing[i] = TRUE;
    Number[i]= Max(Number[0],…Number[n-1])+1;
    Choosing[i] = FALSE;
    for(j=0; j<n; j++){
      while(Choosing[j]) ;
      while((Number[j]!=0)&&((Number[j],j)<(Number[i],i)));
    }

      // critical section
    Number[i]=0
      // remainder section
    } while (TRUE);
```

- Is it possible that many processes receive the same number?

Ans.

Yes，因為可能多個 process 同時執行到這一行，很可能取到一樣的 **Max(Number[0],…Number[n-1])**，所以在後面的條件判斷的地方 **while((Number[j]!=0)&&((Number[j],j)<(Number[i],i)));** 多加了 **i** 和 **j** 做為比較的條件就是為了這種情況發生的時候。

- If we remove the first while loop (i.e. **while(Choosing[j]);**), the mutual exclusion is still maintained?

Ans.

如果少了那一行，可能發生有人還沒抽到籤就開始執行 process 了，假設有一個人已經抽完籤但是還沒 assign 到他的 **Number[i]** 中就被 context switch 走了，等到所有其他人都比完，而且有人進到 critical section 了以後才 switch 回到這個人這裡，一比之下發現自己是最小的可以進去以後，又進去 critical section 了，這時就發生違反 mutual exclusion 的情況了。

- Please prove the correctness of Bakery algorithm in terms of mutual exclusion, progress, and bounded waiting.

Ans.

mutual exclusion:

當一個 process 已經進去 critical section 時，其他 process 會被卡在 **((Number[j],j)<(Number[i],i))** 這條件，所以符合 mutual exclusion。

progress:

如果 Pj 不想進去但 Pi 想進去，Pj 的 **Choosing[i] = FALSE** 且 **Number[j]!=0**，Pi 並不會在 **while** 的地方被 Pj 擋住，而且因為大家都是領完號碼牌照順序進去的，所以不會決策時間過長。

bounded waiting:

都是按照 **Number** 進去 critical section 的，所以不會有插隊問題，也就一定會在有限時間內進得去，符合 bounded waiting。

3. Please implement mutex by using **compare_and_swap**

Ans.

```
do {
    while (!compare_and_swap(&available,true,false));
    /*critical section*/
    available = true;
    /*remainder section*/
} while (true);
```

4. Please describe if busy waiting is eliminated in semaphore implemented by using **block()** and **wakeup()** system calls.

Ans.

是的，用了 **block()** and **wakeup()** 以後，還不符合條件的 process 會被丟到 **waiting queue** 裡面，就不會繼續 busy waiting 了，直到有人做 signal 以後才會再到 **waiting queue** 裡面把 process 抓回來。