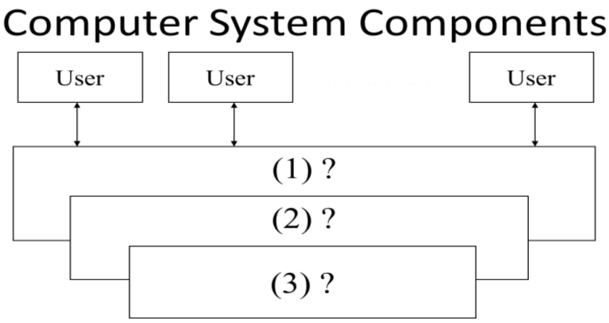
長庚大學104學年度第一學期作業系統期中測驗（滿分112）

系級:　　　　姓名:　　　　學號:

1. (6%) 下圖為一般電腦系統之組成，包含Users、Operating System、Application Programs、Hardware。請分別填入(1)、(2)、(3)對應位置內的內容。



Answer: (1) Application Programs (2%), (2) Operating System (2%), (3) Hardware (2%)

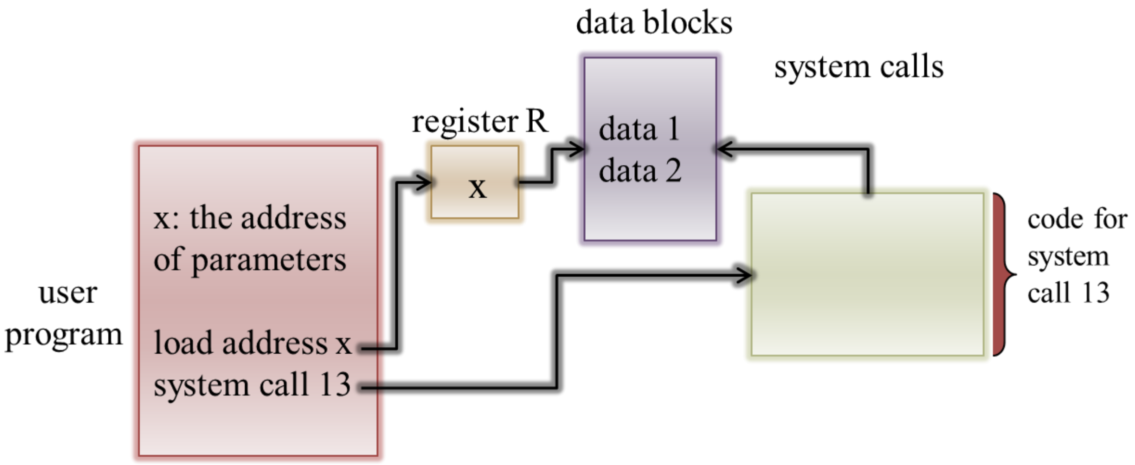
2. (8%) System call的種類有很多，譬如說其中一種是用於Process Control。請舉出另外至少兩種system call的種類。

Answer: (4% for each correct answer)

- File Management
- Device Management
- Information Maintenance
- Communications
- Protection

3. (8%) 應用程式在呼叫system call的時候會需要傳參數給作業系統，傳參數的方法有三種。第一種是用registers：應用程式把參數存在registers裡後再呼叫system call，作業系統便可直接從registers裡讀出參數。第二種做法是用stacks：應用程式把參數push進stacks裡後再呼叫system call，作業系統便可從stacks裡pop出所需的參數。第三種做法是用registers pointing to blocks，請解釋如何使用registers pointing to blocks來傳遞參數。

Answer:



4. (8%) 請定義I/O-bound process與CPU-bound process。

Answer: I/O-bound process – spends more time doing I/O than computations. (5%)
CPU-bound process – spends more time doing computations. (5%)

5. (10%) 在作業系統中我們需要processor schedulers，來協助管理系統中的processes。請定義long-term scheduler (or job scheduler)以及short-term scheduler (or CPU scheduler)的主要功能為何？

Answer: Long-term scheduler – selects which processes should be brought into the ready queue.
Short-term scheduler – selects which process should be executed next and allocates CPU.

6. (8%) 當我們在伺服器上設計網服務程式(如：網頁伺服器、FTP伺服器)，一般來說我們會用multiple threads而不是multiple processes來服務多位使用者。請問，相較之下使用multiple threads的優點為何？

Answer: (Only one correct reason is required)
Threads can share resources of a process, e.g., global data, binary code and opened files. Thus, it is much more efficient in terms of resource saving.
Commutation among the threads of a process is easier than that among process.

7. (10%) 請說明Thread-Local Storage (TLS)的用途，並說明TLS與local variable有何不同。

Answer: Purpose: TLS allows each thread to have its own copy of data.(5%)
Difference: Local variables are visible only during single function invocation, but TLS visible across function invocations in a thread. (5%)

8. (10%) 在使用同質多處理器(Homogeneous processors)時有兩種使用的策略，分別是Asymmetric multiprocessing以及Symmetric multiprocessing。請分別說明這兩種方式是如何使用多處理器。

Answer: Asymmetric multiprocessing – only one processor accesses the system data structures, alleviating the need for data sharing (5%)
Symmetric multiprocessing (SMP) – each processor is self-scheduling, all processes in common ready queue, or each processor has its own private queue of ready processes (5%)

9. (14%) 請寫出以下程式在POSIX環境下執行後的輸出結果。

```
#include<sys/types.h>
#include<stdio.h>
#include<unistd.h>
int main()
{
        pid_t pid, pid2;
        pid = fork();
        if (pid == 0)
        {
                printf("Hello\n");
                pid2 = fork();
                if (pid2 != 0)
                {
                        wait(NULL);
                        printf("Hi\n");
                }
                else
                {
                        printf("Hola\n");
                }
        }
        else
        {
                wait(NULL);
                printf("Bonjour\n");
        }
        printf("Guten tag\n");
        return 0;
}
```

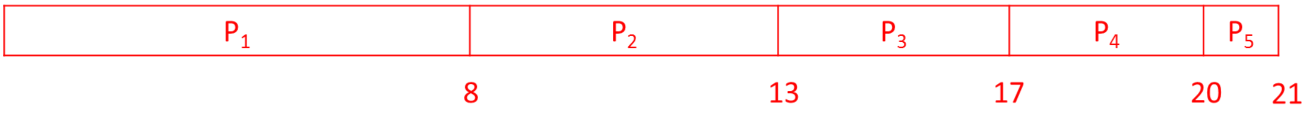
Answer: (-4 for each error)
Hello
Hola
Guten tag
Hi
Guten tag
Bonjour
Guten tag

10. (18%) 考慮已經就緒的五個工作，依序為P_1, P_2, P_3, P_4, P_5。使用三個排程演算法FCFS (First-Come, First-Served)、SJF (Shortest-Job-First)以及RR (Round Robin)來排程，而RR所使用的time quantum為2ms。(1)請畫下三個排程演算法的排程圖，(2)請分別算出三個排程演算法中每個工作的等待時間，若無算式一率不給分(算式可以只是簡單的加減法運算)，(3)請分別算出三個排程演算法的平均等待時間，若無算式一率不給分。

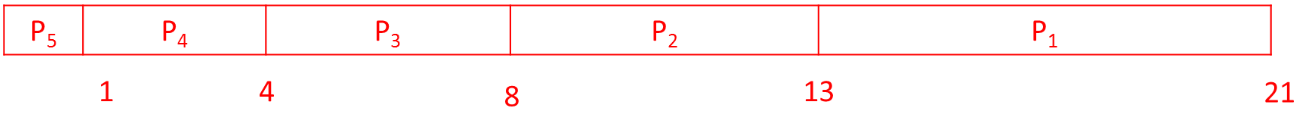
Process	Burst Time
P_1	8 ms
P_2	5 ms
P_3	4 ms
P_4	3 ms
P_5	1 ms

Answer:
(1) (6%)

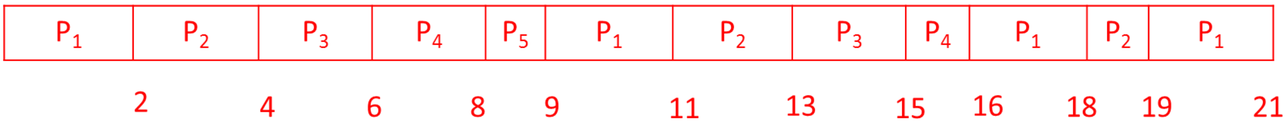
FCFS:



SJF:



RR



(2) (6%) (一定要有算式才給分)

FCFS:	P_1: 8-8=0,	P_2: 13-5=8,	P_3: 17-4=13,	P_4: 20-3=17,	P_5: 21-1=20
SJF:	P_1: 21-8=13,	P_2: 13-5=8,	P_3: 8-4=4,	P_4: 4-3=1,	P_5: 1-1=0
RR:	P_1: 21-8=13,	P_2: 19-5=14,	P_3: 15-4=11,	P_4: 16-3=13,	P_5: 9-1=8

(3) (6%) (一定要有算式才給分)

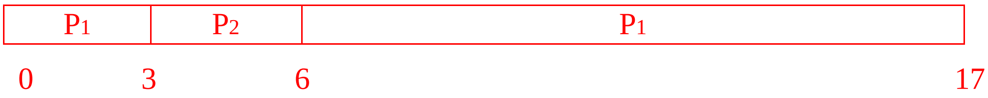
FCFS: (0+8+13+17+20)/5 = 11.6ms SJF: (13+8+4+1+0)/5 = 5.2ms RR: (13+14+11+13+8)/5 = 11.8ms

11. (12%) 有兩個工作P_1及P_2，所需的執行時間(Burst Time)分別是14與3，P_1於時間0到達，P_2於時間3到達，現在考慮兩個排程演算法Preemptive SJF以及Non-preemptive SJF。(1)請畫下兩個排程演算法的排程圖，(2)請分別算出兩個排程演算法的平均等待時間，若無算式一率不給分。

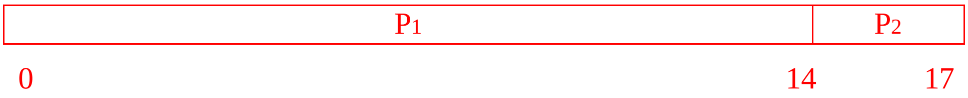
Answer:

(1) (6%)

Preemptive SJF:



Non-preemptive SJF:



(2) (一定要有算式才給分)

Preemptive SJF: ((17-14) + (6-3-3))/2=1.5

Non-preemptive SJF: ((14-14) + (17-3-3))/2=5.5