11120IIS 500500 Linux 作業系統核心 期中考試卷　學號 11006261 9 姓名 楊淨雪

一、是非題(20%)

1. (O) Linux 是一個非商業化的產品。
2. (O) Linux Kernel 裡面包含了常見工具如 ls、mv、cp。
3. (X) Linux 可以給每一個 process 動態大小的 stack。
4. (X) Linux Kernel 使用浮點數跟平常寫 C 程式一樣。
5. (O) GNU C 中的 inline 在編譯過程中會在使用的位置上將函式展開。
6. (O) Linux Kernel 對於條件判斷的最佳化可以使用 likely()與 unlikely()來進行最佳化。
7. (O) 下面程式是執行 rdtsc 組合語言指令，並回傳 timestamp (tsc)暫存器
   unsigned int low, high;
   asm volatile("rdtsc" : "=a" (low), "=d" (high));
8. (O) Unix 系統中所有的東西都被當作檔案來處理，像是硬碟，印表機，USB 等等。
9. (O) Unix 系統中的通常透過 open()、read()、write()、lseek()和 close()來進行裝置或檔案的操作。
10. (O) Linux 上實現了 Unix 的 API 是由 POSIX 標準和其他 Single Unix Specification 定義的。

二、單選題(72%)

1. (A) 哪一個不是 Linux kernel 的 scheduler 策略(A)極小的 time slice(B)process 反應時間短 (C)最大系統使用率(D) scheduler 演算法複雜度越低越好。
2. (A)以下何者錯誤(A)user-space 的 process 為了方便可以直接存取 kernel 內的記憶體資料 (B)user-space 的 process 可以透過 system call 來取得 kernel 中的資料。
   (C)copy_from_user()與 copy_to_user()可以預防 user-space 的程式存取未被授權存取的記憶體空間(D)system call 在 x86-32 系統中是透過 ebx,ecx....暫存器來傳遞參數。
3. (D) Linux 的 scheduler 下面何者錯誤(A)CFS 分為 SCHED_NORMAL 與 SCHED_OTHER(B)Real-time scheduler 的策略分為 SCHED_FIFO 與 SCHED_RR(C)nice 值可以讓使用者自及調整，範圍為-20 到+19 (D)nice 值越高優先權越高。
4. (D)Linux kernel 沒有提供哪種資料結構(A)Link list (B)Queue (C)Map (D) Graph
5. (D) Kernel preemption 不會發生在(A)當 interrupt handler 結束，返回 kernel-space 前 (B)如果 task 在 kernel 直接呼叫 schedule()(C)如果 task 在 kernel 被 block(D)當 system call 返回 user-space。
6. (C)下面何者不是 Linux scheduler 實現的部分(A)Time Accounting (B)Process Selection (C)Interrupt (D)Sleeping and Waking up。
7. (D)關於 CFS 的描述何者錯誤(A)使用 vruntime 來記錄 process 到底運行了多久(B)用紅黑樹來找尋有最小 vruntime 的 process(C)更新 vruntime 是透過 update_curr()由系統計時器呼叫 (D)其演算法複雜度為 O(N)。
8. (D)建立 threads 時所呼叫 clone 所傳進去的 flags 不包含哪一個(A) CLONE_VM (B)CLONE_FS (C)CLONE_FILES (D) CLONE_VFORK。
9. (D) 對於 Linux Kernel 描述以下何者錯誤 (A) 非商業化產品 (B)授權採用 General Public License (C)所有程式碼都是開源的 (D)不得作為商業使用。
10. (D) Linux 系統中，運行中的應用程式不可以透過甚麼方式來跟 Kernel 溝通 (A) system call (B) 讀寫一個 kernel module 對應的 character device (C)讀寫/proc/下的檔案 (D)直接存取 Kernel 中的記憶體位置。
11. (D) Linux Kernel 以下說明是錯誤的(A)模組化的 (B)支援動態載入模組 (C)支援 preemptive 模式 (D) 可同時運行多個 Kernel。
12. (D) Linux Kernel 設定方式不包含以下哪種(A)make config (B)make menuconfig (C)make oldconfig (D)make newconfig

13.(D)下面何者敘述錯誤(A) Linux Kernel 中的 fork()實際上是由 clone()的 system call 來實現的 (B) 當 process 透過 exit() system call 退出時，會將 process 所占用的資源釋放掉(C)當 parent process 比 child process 提早結束，child process 結束時狀態會變成 zombie (D)child process 不能再呼叫 fork()

14.(B) 更換 Linux Kernel 時若沒有執行 make install 會發生哪些狀況(A)無法正常進入系統(B)只能看到舊 kernel 版本的編號(C)kernel 無法正常載入(D)以上皆非。

15.(A) Linux Kernel 內無法使用哪個函數(A)printf() (B)fork() (C)switch_mm() (D) printk()

16.(A) 對於 Linux Kernel 中的 system call 說明何者錯誤(A)具有上萬個 system call (B)具有明確的設計目的 (C)在 kernel-space 中運行 (D)已經存在的 system call 不會任意改變規格

17.(A) Unix 的 Process 建立是透過(A)fork() (B)copy() (C)write() (D)vcopy() 來完成

18.(A) Process 不會有以下哪一種狀態(A)TASK_BLOCK (B)TASK_RUNNING (C) TASK_INTERRUPTIBLE (D) TASK_UNINTERRUPTIBLE

19.(A)下列程式中的 task 可以(A)取得目前 process 的每一個 child process 的 task_struct (B)可以取得系統中每一個 process 的 task_struct(C)只可以取得 parent process 的進入點 (D)取得目前的 process 的進入點。

```
struct task_struct *task;
struct list_head *list;
list_for_each(list, &current->children) {
   task = list_entry(list, struct task_struct, sibling);
}
```

20.(B)下面程式中的 task 在執行後會獲得(A) 0x0 (B) init process 的 task_struct (C) 目前 process 的 task_struct (D) 目前 process 的 parent 的 task_struct。

```
struct task_struct *task;
for (task = current; task != &init_task; task = task->parent);
```

21.(D)Linux kernel 中下面何者錯誤(A)在 fork()的時候是採用 copy-on-write (B) 呼叫 fork()後立刻使用 exec()載入程式不需要複製 parent process 所使用的資源(C) 當 process 透過 exit() system call 退出時，會將 process 所占用的資源釋放掉(D)process description 是用一個 queue 的結構存放。

22.(D) process fork 的過程中以下何者錯誤(A)可以用 alloc_pid()來替 child process 來向 kernel 索取一個新的 PID (B)系統可以接受的 PID 最大值可以透過 cat /proc/sys/kernel/pid_max 取得 (C)完成 dup_task_struct()後應該要先將 process 的狀態設定為 TASK_UNINTERRUPTIBLE (D)PF_FORKNOEXEC 表示會另外呼叫 exec()

23.(D)下面哪種程式不是 processor-bound(A)ssh-keygen (B)MATLAB (C)machine learning (D)Word。

24.(A) process 終止時不需要呼叫下面哪一個(A)unlock_task() (B)exit_mm() (C)exit_notify() (D)acct_update_integrals

三、問答題(8%) (可直接使用程式碼或是用描述的方式說明你的**輸出輸入機制**)

有一硬體設備為 8 個 RGB 光源每個顏色有 256 階變化，需由對應的記憶體來改變燈源顏色，控制卡可插於主機板 ISA 介面中，光源顏色狀態會對應到記憶體的開始位置為 0xC0000，每一個光源由 3 個 byte 控制，順序分別為 RGB。Kernel 中已有 API 可存取記憶體位置資料

讀取資料 unsigned int val=isa_readl(addr) ， 寫入資料 boolean ret=isa_writel(val,addr)

設備廠商希望幫他設計一個驅動程式可以透過/proc/RGB 來調整燈號狀態，並可以個別調整每一個 RGB 光源調整顏色。請說明你的設計針對/proc/RGB 的輸出入方式或寫下 pseudo code。

一.

1. O
2. ~~O~~ -2
3. X
4. X
5. O
6. O
7. O
8. O
9. O
10. O

二.

1. A 2. A 3. D 4. D 5. D
6. C 7. D 8. D 9. D 10. D
11. D 12. D 13. D 14. B 15. A
16. A 17. A 18. A 19. A 20. B
21. D 22. D 23. D 24. A

三.

即 0xC0000 開始的 8個光源
↑ 每個光源用 3 bytes 控制

read(): 透過 unsigned int val = isa_read1(addr) 讀取 光源顏色狀態，
再由 copy_to_user() 將此 val 複製到 user space，完成讀取。

write(): 透過 copy_from_user() 將 光源顏色狀態 從 user space
/proc / RGB
複製到 kernel space，並根據 boolean ret = isa_write1(val, addr) 來判斷
是否成功寫入硬體設備.

- 個別調整方式（請見下頁）