作業八:

學習目標:

- 在傳統 UNIIX 系統中，權限只有二種「全給」或「受限」，在比較新的 Linux 支援「capabilities」，可以將 super user 的權限「部分」給予某個應用程式，藉由 capabilities，可以讓 Linux 的權限設定更為細緻，進而增加系統的安全性。
- 例如：chown_super 的「能力」只有「允許任何可以執行這個檔案的人，改變任意檔案的 owner」，而不是「允許任何可以執行這個檔案的人，擁有所有權限，包含改變檔案的 owner」。
- 這可以避免 chown_super 被駭客攻擊時，駭客從 chown_super 得到控制系統的所有權限。
- 了解 nice 的在 CPU scheduling 上的用法

先練習:

執行下面命令，黃色底的部分是註解

$ cp /usr/bin/chown ./chown_super 將 chown 複製到當前目錄下

$ sudo setcap CAP_CHOWN+ep ./chown_super 讓 chown_super 擁有更改任意檔案的 owner 的權利。權利的選項可以 man capabilities 查看 http://man7.org/linux/man-pages/man7/capabilities.7.html

這句話的意思是：因為 setcap 需要用超級使用者的權限設定，因此用 sudo 執行，然後賦予 chown_super change owner 的權限（即：CAP_CHOWN），而這個權限的賦予方式是:

Permitted（強制賦予）：無論這個 process 的老爸是誰，都立即賦予他這項權限

Effective：執行這個檔案的時候，所設定的權限有效（我知道這個很怪，但如果設定檔案的「能力」，「e」一定要有）（這個屬性在設定 task 時（我們還沒教怎樣建立 task）的意義才會明確）

$ ./chown_super YOUR_USER_NAME /bin/ls 將/bin/ls 的檔案的 owner 變更為「你自己」

$ ./chown_super root /bin/ls 再將/bin/ls 的檔案的 owner 變回 root

題目:

將 nice 複製到自己的目錄下，名為 nice_pro，必且讓 nice_pro 擁有提高優先權的能力

舉例:

還未設定前：

shiwulo@vm:~$ ./nice_pro -n -10 ls　執行「ls」時將優先等級提高 10

./nice_pro: cannot set niceness: Permission denied

a chown chown_super downloads files git kill_super nice_pro snap strace_pro workspace

設定後（不會出現權限不足的問題）

shiwulo@vm:~$ ./nice_pro -n -10 ls

a chown chown_super downloads files git kill_super nice_pro snap strace_pro workspace

提示：複製檔案到自己的目錄

whereis nice　//找到 nice 在哪裡，然後 cp 過來

shiwulo@vm:~/downloads$ whereis nice

```
nice: /usr/bin/nice /usr/share/man/man1/nice.1.gz
/usr/share/man/man2/nice.2.gz
shiwulo@vm:~/downloads$ cp /usr/bin/nice nice-pro
shiwulo@vm:~/downloads$ cp /usr/bin/nice nice-pro-2
shiwulo@vm:~/downloads$ sudo setcap CAP_SYS_NICE+ep ./nice-pro
shiwulo@vm:~/downloads$ sudo chown root:root ./nice-pro-2
shiwulo@vm:~/downloads$ sudo chmod +s ./nice-pro-2
shiwulo@vm:~/downloads$ nice -n -10 ls
nice: cannot set niceness: Permission denied
code_1.54.3-1615806378_amd64.deb  linux-5.12.tar.xz  nice-pro2
imager_amd64.deb                  nice-pro          ubuntu-20.10-preinstalled-server-arm64+raspi.img
linux-5.12                        nice-pro-2        ubuntu-20.10-preinstalled-server-arm64+raspi.img.xz
shiwulo@vm:~/downloads$ ./nice-pro -n -10 ls
code_1.54.3-1615806378_amd64.deb  linux-5.12.tar.xz  nice-pro2
imager_amd64.deb                  nice-pro          ubuntu-20.10-preinstalled-server-arm64+raspi.img
linux-5.12                        nice-pro-2        ubuntu-20.10-preinstalled-server-arm64+raspi.img.xz
shiwulo@vm:~/downloads$ ./nice-pro2 -n -10 ls
code_1.54.3-1615806378_amd64.deb  linux-5.12.tar.xz  nice-pro2
imager_amd64.deb                  nice-pro          ubuntu-20.10-preinstalled-server-arm64+raspi.img
linux-5.12                        nice-pro-2        ubuntu-20.10-preinstalled-server-arm64+raspi.img.xz
```

討論一：

使用 chmod +s 會給 nice-pro2 所有 super-user 的權限，萬一 nice-pro2 有安全性漏洞，那麼這個執行檔案被破解，駭客就拿到 super user 的權限

討論二：

sudo setcap CAP_SYS_NICE+ep 只是給 nice-pro 有提高優先權的權利，萬一 nice-pro 有安全性漏洞，那麼這個執行檔案被破解，駭客只拿到提升優先權的權限

題目二：

想辦法量測 nice 提高優先權的比例

什麼是 nice?

舉例：學長姊你人很 nice，但我比較喜歡 S P強者

翻譯：nice 指的是『會禮讓別人』，因此越 nice（nice 的值越大），相當於搶奪資源的權力越小

自己可以將自己的 nice 提高（提高禮讓程度），例如：nice -n 5 ls

但要把自己的 nice 降低（降低禮讓程度）需要有特殊權限：例如：nice -n -5 ls

撰寫一個會跑很久的程式

```
int main() {
        for (int i=0; i<1000000000; i++)
                ;
}
```

執行看看

```
shiwulo@sp1:~$ time nice -n 0 ./a.out

real 0m1.582s
user 0m1.578s
sys  0m0.004s
shiwulo@sp1:~$ time nice -n +10 ./a.out

real 0m1.587s
user 0m1.583s
sys  0m0.005s
```

小結論：幾乎沒有差別，這是因為系統的負載太輕。nice 是相對的。如果系統上只有一個 process 在執行 nice 的差別不大

```
//鬧鐘的時間到了，要做什麼事情?
```

```
void alarmHandler(int signo) {
        printf("cpp = %lld\n", cpp);
        exit(0);
}

int main(int argc, char** argv) {
        int nice_v=atoi(argv[1]); //讀入 nice 的參數
        int childPid = fork();   //產生二個行程
        if (childPid > 0) {       //養我們的雙親
                nice(nice_v);     //修改 parent 的 nice 值
        } else {
                //child 不用特別處理
        }
        //底下的程式碼無論 child 是否大於 0 都會執行
        //設定鬧鐘（SIGALRM）叫的時候，『作業系統』呼叫
alarmHandler
        signal(SIGALRM, alarmHandler);
        //把鬧鐘的時間定在 1 秒鐘以後
        alarm(1);
        //不斷地 cpp++
        while (1) {
                cpp++;
        }
}
```

結果是:

```
shiwulo@sp1:~$ ./a.out 5
cpp = 650929577
cpp = 642166639
```

小結論：應該是因為多核心，因此 OS 自動將 parent 和 child 分配到不同顆 core 上執行

使用 taskset 將 child 和 parent 固定在同一顆處理器

```
shiwulo@sp1:~$ taskset 0x1 ./a.out 5
cpp = 1575378834
cpp = 4832648534
shiwulo@sp1:~$
```

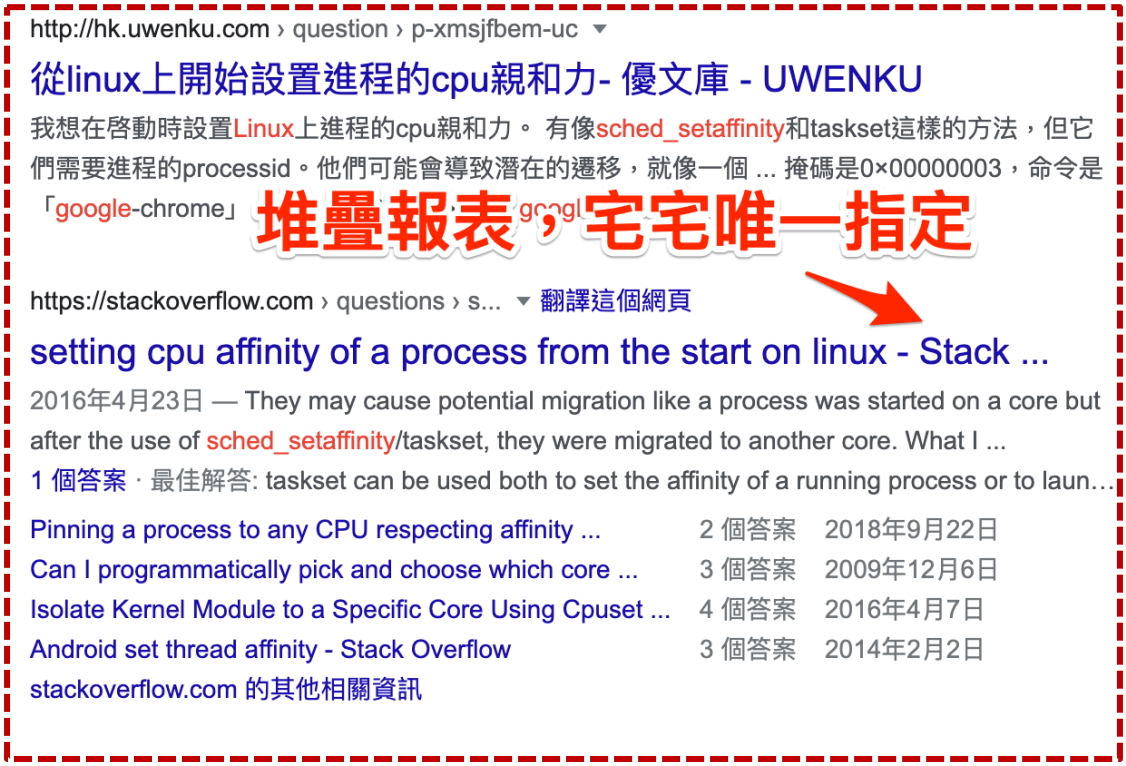
速度約差了 3.06 倍。nice 每一個等級相差 1.25 倍，$1.25^5 = 3.05$

到這裡作業的程式部分

自行修煉，寫出程式碼能自己綁定處理器的程式

提示:

1. google setaffinity linux




```
//以下的程式碼可以試試看在 fork 前和 fork 後設定有何差別
cpu_set_t mask;   //CPU 核的集合
CPU_ZERO(&mask);
CPU_SET(1,&mask);     //先做好參數設定，綁在第一顆處理器
sched_setaffinity(0, sizeof(mask), &mask)
```

查到大致的方向以後，再查一下 man page

```
CPU_SET(3)                Linux Programmer's Manual                CPU_SET(3)

NAME
       CPU_SET,  CPU_CLR,  CPU_ISSET,  CPU_ZERO, CPU_COUNT, CPU_AND, CPU_OR,
       CPU_XOR, CPU_EQUAL, CPU_ALLOC, CPU_ALLOC_SIZE,  CPU_FREE,  CPU_SET_S,
       CPU_CLR_S, CPU_ISSET_S, CPU_ZERO_S, CPU_COUNT_S, CPU_AND_S, CPU_OR_S,
       CPU_XOR_S, CPU_EQUAL_S - macros for manipulating CPU sets
```

```
SYNOPSIS
        #define _GNU_SOURCE                 /* See feature_test_macros(7) */
        #include <sched.h>

        void CPU_ZERO(cpu_set_t *set);

        void CPU_SET(int cpu, cpu_set_t *set);
        void CPU_CLR(int cpu, cpu_set_t *set);
        int  CPU_ISSET(int cpu, cpu_set_t *set);
```

```
SCHED_SETAFFINITY(2)        Linux Programmer's Manual        SCHED_SETAFFINITY(2)

NAME
        sched_setaffinity,  sched_getaffinity  -  set  and get a thread's CPU
        affinity mask

SYNOPSIS
        #define _GNU_SOURCE                 /* See feature_test_macros(7) */
        #include <sched.h>
        int sched_setaffinity(pid_t pid, size_t cpusetsize, const cpu_set_t *mask);
        int sched_getaffinity(pid_t pid, size_t cpusetsize, cpu_set_t *mask);
         sched_setaffinity() sets the CPU affinity mask of the thread whose ID is pid to the
         value specified by mask.  If  pid  is  zero,  then  the calling  thread  is  used.
         The argument cpusetsize is the length (in bytes) of the data pointed to by mask.
         Normally this argument  would be specified as sizeof(cpu_set_t).
```

報告:

1. 報告上面寫上姓名（可隱匿一個字）和學號
2. 從 man capabilities 裡面隨便挑三個權限，並說明那三個權限是什麼樣的用途（大致上就是英文翻譯成中文再加上一點點自己的理解）
3. 想辦法量測『優先權高一等級的 task 比正常優先權的 task 速度快多少』？

繳交:

1. 程式碼和 makefile，助教執行『sudo make』指令後，必須自動產生 nice_testing，nice_testing 要自動 fork 出一個 child，child 和 parent 的優先等級相差 5。
2. 撰寫報告，格式並須為 pdf。測試報告前請附上姓名（可隱匿一個字）

及學號

3. 請將所有檔案壓縮成.tar.bz2。繳交到 ecourse2 上
4. 不能遲交
5. 再次提醒，助教會將所有人的作業於 dropbox 上公開
6. 繳交期限：2021/5/11 早上八點
7. 如果真的不會寫，記得去請教朋友。在你的報告上寫你請教了誰即可。