作業九：

學習目標：

Linux 建立 task 的方式與 UNIX 相同，使用 fork()函數。fork 建立的 child task 與 parent 幾乎一模一樣。尤其是『程式碼』是一樣的。

這種看似奇怪的方式與 UNIX 當初的設計目的有關，UNIX 主要是作為伺服器。以 web server 而言，收到新的 socket 連線時，web server 建立一個新的「執行體」，這個執行體功能與原本的 web server 一模一樣，只是新的執行體只會服務「這個新的連線」。

題目：

- 寫一隻小的應用程式名稱為 mylogin，每當使用者輸入姓名時，mylogin 會判斷這個使用者是否可以進入伺服器，判斷的依據為該成員是否在/etc/passwd 內。『不用輸入密碼』
- 如果可以進入伺服器，則 login 會產生一個 child process，這個 child process 會設定適當的變數，然後執行『bash』
- 當使用者離開 shell 以後，要再跳出提示符號，讓使用者再次登入
- 挑戰：是否可以讓使用者輸入密碼，然後到/etc/shadow 內驗證密碼呢？（不計分）

報告：

1. 此次作業不需要繳交報告

繳交：

1. 程式碼和 makefile，助教執行『make』指令後，必須自動產生 mylogin。
   甲、
2. 請將所有檔案壓縮成.tar.bz2。繳交到 ecourse2 上
3. 不能遲交
4. 再次提醒，助教會將所有人的作業於 dropbox 上公開
5. 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。

關於程式碼：

```
#include <stdlib.h>
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
#include <pwd.h>
#include <assert.h>
```

```c
#include <string.h>
#include <ctype.h>
#include <grp.h>
#include <limits.h>


char* ltrim(char *s)
{
    while(isspace(*s)) s++;
    return s;
}

char* rtrim(char *s)
{
    char* back = s + strlen(s);
    while(isspace(*--back));
    *(back+1) = '\0';
    return s;
}

char *trim(char *s)
{
    return rtrim(ltrim(s));
}

int main(int argc, char* argv[]) {
    char username[1024];
    char* namePtr;
    //<limits.h>
    char passwordStr[sysconf(_SC_PASS_MAX)];
    struct passwd passwd_ent;
    struct passwd *result;
    struct group *gr;
    char buffer[1024];
```

```
    long ngroups_max;
    gid_t gid;
    gid_t groups[sysconf(_SC_NGROUPS_MAX)];
    int nGroup = sysconf(_SC_NGROUPS_MAX);
    int ret;

    // 🔴 🟠 🟡 🟢 🔵 🟣
    // 這裏使用 label 加上 goto 的原因如下
    // 1. goto + label 是有名字的，名字就是 label
    // 2. 如果使用 while(1)，會造成 body 太過長，而我又不太想把程式碼分成小函數
    // 3. 綜合上面所述， goto 比 while?(1) 的可讀性要來得好，因此我使用 goto
    // 4. 就像是我註解喜歡用 😅 😂 🤣『唐詩』＋emoji 😅 😂 🤣

relogin :
    printf("請輸入名稱\n");
    //assert(fgets(username, 1024, stdin)!=NULL);
    namePtr = fgets(username, 1024, stdin);
    printf("gets %s\n", namePtr);
    printf("請輸入密碼\n");
    // 🔴 🟠 🟡 🟢 🔵 🟣
    // 獲取使用者輸入的密碼，但我後續沒有驗證密碼，
    strncpy(passwordStr, getpass("請輸入密碼"), sysconf(_SC_PASS_MAX));

    //將字串前後的非 ASCII 的符號去掉
    namePtr = trim(namePtr);

    //int getpwnam_r(const char *name, struct passwd *pwd,
```

```
    //char *buffer, size_t bufsize, struct passwd
**result);
    //查詢這個使用者是否在作業系統中
    ret = getpwnam_r(namePtr, &passwd_ent, buffer,
1024, &result);
    if (ret != 0)
    {
        perror("發生錯誤，必須吐一些東西到螢幕上：");
        goto relogin;
    }
    // 🔴🟠🟡 應該在這個地方使用 fork 🔴🟠🟡
    //查詢這個使用者還屬於哪些 group
    ret = getgrouplist(namePtr, passwd_ent.pw_gid,
groups, &nGroup);
    printf("getgrouplist = %d\n", ret);
    printf("使用者編號: %d\n", passwd_ent.pw_uid);
    printf("使用者名稱: %s\n", passwd_ent.pw_name);
    printf("群組編號：%d\n", passwd_ent.pw_gid);
    printf("家目錄: %s\n", passwd_ent.pw_dir);
    printf("其他訊息 %s\n", buffer);
    printf("所隸屬的所有群組： ");
    printf("共%d 個\n", nGroup);
    for (int i=0; i< nGroup; i++) {
        gr = getgrgid(groups[i]);
        printf("%s, ", gr->gr_name);
    }
    printf("\n");

    //int setgroups(size_t size, const gid_t
*list);
    //setgroups() sets the supplementary group IDs
for the calling process.
```

```
    //On success, setgroups() returns 0.  On error,
-1 is returned, and errno is set appropriately.
    // 🔴 🟠 🟡 🟢 🔵 🟣
    //先設定使用者的 gid
    assert(setgid(passwd_ent.pw_gid)==0);
    // 🔴 🟠 🟡 🟢 🔵 🟣
    //改變工作目錄，避免使用者一開始的目錄不是家目錄，讓初階
的使用者感覺很怪
    //此外避免有些應用程式在初始化時，把工作目錄當成家目錄
    assert(chdir(passwd_ent.pw_dir)==0);
    // 🔴 🟠 🟡 🟢 🔵 🟣
    //int setenv(const char *name, const char
*value, int overwrite);
    // 🔴 🟠 🟡 🟢 🔵 🟣
    // 改變環境變數"HOME"，照理說應該要依照使用者的設定給最
基本的環境變數，但我很懶沒做
    setenv("HOME", passwd_ent.pw_dir, 1);
    //A process can drop all of its supplementary
groups with the call
    //setgroups(0, NULL);
    // 🔴 🟠 🟡 🟢 🔵 🟣
    // 改變使用者的 UNIX 群組
    setgroups(0, NULL);
    // 改變使用者的擴充群組
    setgroups(sysconf(_SC_NGROUPS_MAX), groups);
    // 🔴 🟠 🟡 🟢 🔵 🟣
    // 上述的權限設定都可以成功是因為目前 uid 還是 root，下一
行執行下去以後，就無法改變權限設定了
    assert(setuid(passwd_ent.pw_uid) == 0);
    // 🧐 🤓 😎 🥸 🤩 🥳
    // 底下就是這次作業的重點，如果使用 system 實現，那麼我
們剛剛改變的都是『這個行程』，但在上一行已經放棄 root 權限
    // 因此當有使用者離開，下個使用者要重新 login，已經無法
改變權限了，所以要先 fork，child 慢慢的拋棄權限
```

```
    // 而 parent 繼續持有 root 權限，這樣才能在下一位使用者
login 時改變權限
    // 把底下這一行改成用 execvp 實現
    // system 其實就是 fork + execvp + wait 實現的
    ret = system("bash");
    printf("bash 的回傳值是 %d\n", ret);
goto relogin;
}
```