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. 繳交期限：2021/5/18 早上八點
6. 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>

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;
        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;

relogin:
        printf("請輸入名稱\n");
```

```
//assert(fgets(username, 1024, stdin)!=NULL);
namePtr = fgets(username, 1024, stdin);
printf("gets %s\n", namePtr);



//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.
        assert(setgid(passwd_ent.pw_gid)==0);


        assert(chdir(passwd_ent.pw_dir)==0);
        //int setenv(const char *name, const char *value, int overwrite);
        setenv("HOME", passwd_ent.pw_dir, 1);
        //A process can drop all of its supplementary groups with the call
        //setgroups(0, NULL);
        setgroups(0, NULL);
        setgroups(sysconf(_SC_NGROUPS_MAX), groups);
        assert(setuid(passwd_ent.pw_uid) == 0);
        //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;
}
```