

```
#include <stdlib.h>
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
#include <pwd.h>
#include <assert.h>
#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;    //舊的UNIX只有一個群組  
    gid_t groups[sysconf(_SC_NGROUPS_MAX)];新的有很多個群組  
    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 bufsz, 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); //
```

查尋到哪一個shell







```
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   

```
int pid;
```

```
pid=fork();
```

```
if(pid>0){ //parent 持續使用root的權限
```

```
int wret;
```

```
wiat(&wret);//等child執行結束
```

```
goto relogin;
```

```
}else //child轉換成一般使用者權限
```

//查詢這個使用者還屬於哪些group

```
ret = getgrouplist(namePtr, passwd_ent.pw_gid, groups, &nGroup);//
```

查尋到哪一個shell

```
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); //後面那個變數的home放到
懷竟變數裡面

//A process can drop all of its supplementary groups with the call

```
//setgroups(0, NULL);
```

```
setgroups(0, NULL); //丟掉目前所有所屬的群組
```

setgroups(sysconf(_SC_NGROUPS_MAX), groups); //設定這個人所
隸屬的群組

```
assert(setuid(passwd_ent.pw_uid) == 0); //改變使用者
```

//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"); //跳到shell
```

```
execvp("bash"); //執行使用者shell
```

```
}  
  
//printf("bash的回傳值是 %d\n", ret);  
  
//goto relogin;  
  
}
```

```
assert(setgid(pwd_ent.pw_gid)==0);
```

第二次錯誤：原本root，在第一輪丟掉權限，只有root可以變成其他人的權限

```
// ●●● 應該在這個地方使用 fork ●●●
```