

Assignment 11

Team18

112062519 廖思愷 112062636 游竣量 111065547 游述宇

一、

```
int main(int argc, char **argv) {
    char cwd[1024];
    if (getcwd(cwd, sizeof(cwd)) == NULL) {
        perror("getcwd error");
        exit(1);
    }

    daemonize("daemonize");

    char *login;
    login = getlogin();

    char filepath[1024];
    FILE *fp;
    sprintf(filepath, "%s/assignment11.txt", cwd);
    fp = fopen(filepath, "w");
    fprintf(fp, "Login name: %s\n", login);
    fclose(fp);

    exit(0);
}
```

1. 調用 `daemonize` 函數來創建守護進程。

```
daemonize("daemonize");
```

2. 使用 `getlogin` 函數來取得 `login name`，接著創建 `assignment11.txt`，並將
“Login name: [login name]” 寫入 `assignment11.txt` 中。

```
char *login;
login = getlogin();

char filepath[1024];
FILE *fp;
sprintf(filepath, "%s/assignment11.txt", cwd);
fp = fopen(filepath, "w");
fprintf(fp, "Login name: %s\n", login);
fclose(fp);

exit(0);
}
```

輸出結果：assignment11.txt

```
Login name: team18  
PID: 20048  
PPID: 1  
Session ID: 20047
```

- 實際 freeBSD 得到 Login name: team18

3. daemonize 函數中的每一個步驟與目的如下：

- 設置文件創建掩碼為 0，確保 daemon 創建的任何文件都有適當的權限。
- 使用 getrlimit 確定系統允許的最大文件描述符數量。使得 daemonize 函數可以遍歷並關閉所有可能的 file descriptor，確保 daemon 不會無意中保持打開不必要的檔。
- 使用 fork 創建 child process，然後使用 setsid 使 child process 成為新的 session leader，從而與 control terminal 分離。
- 當會話組長終止時，會發送 SIGHUP 信號到它的子進程。在這裡將 SIGHUP 的處理方式設為忽略，確保守護進程在會話組長結束後不會被終止。
- 再次進行 fork，確保進程不是 session leader process，使之無法打開新的控制終端。
- 切換到 root 目錄，防止 daemon 阻止其他文件系統被卸載。
- 關閉所有打開的文件描述符，釋放不再需要的資源。
- 把 stdin、stdout 和 stderr，redirect 到 /dev/null，以確保 daemon 在後台運行時不會嘗試讀取輸入或產生輸出，這樣可以避免潛在的 hang 和資源浪費，同時保護 daemon 不受使用者交互的干擾。
- 使用 openlog 初始化 log file 用於確認 error report 和其他資訊。

4. 從上述 daemonize function 的功能可以得知，在 process 成為 daemon 後，會有幾個主要的改變：

- 後台運行：daemon 在後台運行，與任何 control terminal 或 user

interface 分離。

- (b) 獨立於 parent：在 daemonize 過程中，原始的 parent 終止，daemon 被重新指派給 init 進程，以確保守護進程獨立於其原始控制終端。
- (c) 關閉 file descriptor：daemon 關閉了大部分或所有從 parent 繼承的 file descriptor（ex. stdin、stdout、stderr）以確保 daemon 不會無意中使用或干擾這些 file descriptor。
- (d) 改變 working directory：守護進程通常將其工作目錄更改為根目錄，以避免鎖定任何正在使用的目錄，這可能會阻止它被卸載。
- (e) 標準 I/O 處理：標準輸入、輸出和錯誤流被 redirect 到 /dev/null 或其他適當的目的地，以防止 daemon 嘗試從終端讀取 input 或向 terminal 寫 output。
- (f) 持續運行：一旦 daemonize，如果有設類似 while 迴圈 process 會持續運行，直到被明確停止或系統關閉。
- (g) 忽略 SIGHUP 信號：daemon 會忽略 SIGHUP 信號，以避免因父進程關閉而終止。