

# TINY SHELL

Phạm Văn Thông

## 1 Giới thiệu

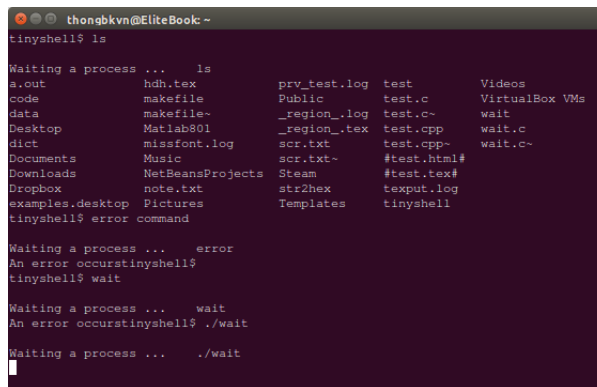
Chương trình tinyshell thực hiện những chức năng cơ bản:

- Khởi tạo tiến trình
- Khởi tạo tiến trình nền (background process).
- Chạy chương trình theo lô

ngoài ra có thể thực hiện những chức năng của shell như hủy tiến trình, liệt kê những tiến trình đang chạy bằng các lệnh `ps`, `kill` nhưng đó không phải là các thành phần của chương trình tinyshell này mà là các chương trình của hệ điều hành.

### 1.1 Khởi tạo tiến trình

Khởi tạo một tiến trình bằng cách gõ câu lệnh sau dấu nhắc `$`. Nếu câu lệnh không được thực hiện, màn hình sẽ hiển thị tinyshell `$ An error occurs.`

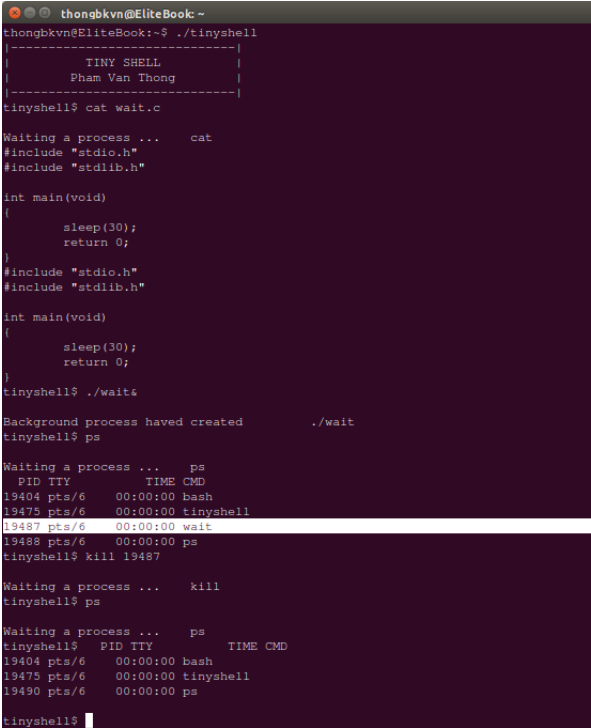


```
thongbkvn@EliteBook ~  
tinyshell$ ls  
Waiting a process ... ls  
a.out      hdh.tex      prv_test.log  test          Videos  
code       makefile     Public        test.c        VirtualBox VMs  
data       makefile-    _region_.log  test.c-       wait  
Desktop    Matlab801    _region_.tex  test.cpp      wait.c  
dict       missfont.log scr.txt       test.cpp-     wait.c-  
Documents  Music        scr.txt-      #test.html#  
Downloads  NetBeansProjects Steam         #test.tex#  
Dropbox    note.txt     str2hex       texput.log  
examples.desktop Pictures      Templates     tinyshell  
tinyshell$ error command  
Waiting a process ... error  
An error occurs  
tinyshell$ wait  
Waiting a process ... wait  
An error occurs  
tinyshell$ ./wait  
Waiting a process ... ./wait
```

Hình 1: Khởi tạo tiến trình.

## 1.2 Khởi tạo tiến trình nền

Để khởi tạo tiến trình nền (tiến trình chạy ẩn- background process). Ngay sau câu lệnh người dùng nhập vào cần thêm kí tự &. Có thể thêm ngay trực tiếp hoặc sau một (vài) dấu cách. Ví dụ như sau:



```
thongbkvn@EliteBook:~$ ./tinysHELL
-----
TINY SHEL
Pham Van Thong
-----
tinysHELL$ cat wait.c
Waiting a process ...   cat
#include "stdio.h"
#include "stdlib.h"

int main(void)
{
    sleep(30);
    return 0;
}
#include "stdio.h"
#include "stdlib.h"

int main(void)
{
    sleep(30);
    return 0;
}
tinysHELL$ ./wait&
Background process haved created      ./wait
tinysHELL$ ps
Waiting a process ...   ps
  PID TTY          TIME CMD
 19404 pts/6        00:00:00 bash
 19475 pts/6        00:00:00 tinysHELL
 19487 pts/6        00:00:00 wait
 19488 pts/6        00:00:00 ps
tinysHELL$ kill 19487
Waiting a process ...   kill
tinysHELL$ ps
Waiting a process ...   ps
  PID TTY          TIME CMD
 19404 pts/6        00:00:00 bash
 19475 pts/6        00:00:00 tinysHELL
 19490 pts/6        00:00:00 ps
tinysHELL$
```

Hình 2: Khởi tạo tiến trình nền

Trong ví dụ trên, chương trình wait thực hiện sleep trong vòng 30 giây rồi kết thúc. Ta sử dụng lệnh `./wait&` (giả sử chương trình nằm ngay cùng thư mục với tinysHELL), sau khi sử dụng lệnh trên, ta có ngay dấu nhắc để thực hiện lệnh tiếp theo mà không cần phải chờ chương trình wait thực hiện xong. Lưu ý rằng, ngay khi chương trình trong quá trình thực hiện có thể hiển thị lên shell và điều đó là không kiểm soát được. Ví dụ chương trình sau đây thực hiện in ra các số từ 1 đến 9, sau mỗi lần in ra nó nghỉ trong vòng 3 giây. Thực hiện chạy nền chương trình, có thể ta sẽ có kết quả như sau:

```
thongbkvn@EliteBook ~  
-----|  
| TINY SHELL |  
| Pham Van Thong |  
|-----|  
tinyshe11$ ./count&  
  
Background process haved created      ./count  
tinyshe11$  
count: 1  
ls  
Waiting a process ...    ls  
a.out    Downloads    NetBeansProjects  Steam    #test.tex#  
code     Dropbox    note.txt         str2hex  texput.log  
count    examples.desktop Pictures         Templates tinyshe11  
count.c  hdh.tex    prv_test.log     test     Videos  
count.c~ makefile   Public           test.c   VirtualBox VMs  
data     makefile~  _region_.log     test.c~  wait  
Desktop  Matlab801  _region_.tex     test.cpp wait.c  
dict     missfont.log scr.txt~         test.cpp~ wait.c~  
Documents Music    scr.txt~         #test.html#  
tinyshe11$  
count: 2  
ps  
Waiting a process ...    ps  
PID TTY      TIME CMD  
19865 pts/9    00:00:00 bash  
19937 pts/9    00:00:00 tinyshe11  
19938 pts/9    00:00:00 count  
19940 pts/9    00:00:00 ps  
tinyshe11$  
count: 3  
  
count: 4  
  
count: 5  
  
count: 6  
  
count: 7  
  
count: 8
```

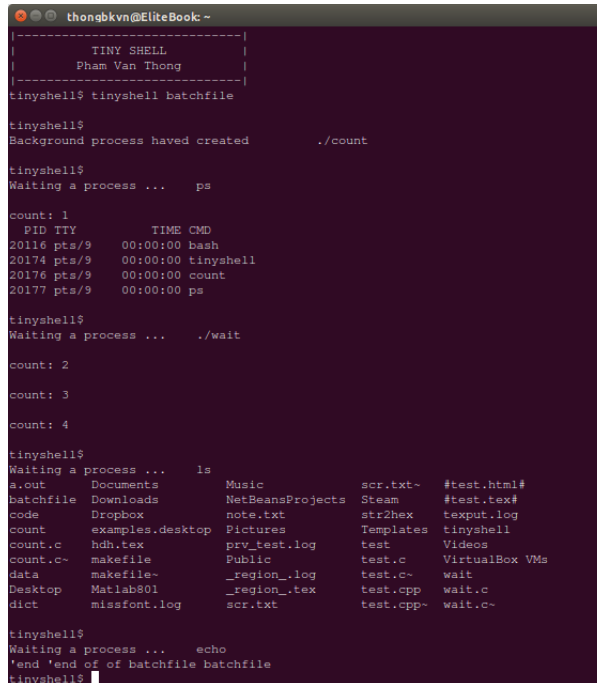
Hình 3: Chương trình nền

### 1.3 Thực hiện theo lô

tinysHELL có thể thực hiện các chương trình theo lô (batch). Thứ tự các chương trình sẽ được thực hiện được lưu tại 1 file văn bản, mỗi câu lệnh được ghi trên một dòng. tinysHELL sẽ tuần tự thực hiện từng câu lệnh từ trên xuống dưới. Nếu gặp một câu lệnh lỗi, nó vẫn tiếp tục thực hiện câu lệnh khác. Để thực hiện các chương trình theo lô, câu lệnh cần có cấu trúc như sau: tinysHELL path/to/batchfile. Sau đây là ví dụ về 1 batchfile.

```
./count&
ps
./wait
ls
echo 'end of batchfile'
```

Và đây là kết quả thực hiện



```
thongbkvn@EliteBook ~
-----
| TINY SHELL |
| Pham Van Thong |
|-----|
tinysHELL$ tinysHELL batchfile

tinysHELL$
Background process haved created      ./count

tinysHELL$
Waiting a process ...    ps

count: 1
  PID TTY          TIME CMD
20116 pts/9        00:00:00 bash
20174 pts/9        00:00:00 tinysHELL
20176 pts/9        00:00:00 count
20177 pts/9        00:00:00 ps

tinysHELL$
Waiting a process ...    ./wait

count: 2

count: 3

count: 4

tinysHELL$
Waiting a process ...    ls
a.out    Documents    Music        scr.txt~    #test.html#
batchfile Downloads    NetBeansProjects Steam      #test.tex#
code     Dropbox      note.txt     str2hex     texput.log
count    examples.desktop Pictures     Templates   tinysHELL
count.c  hdh.tex      prv_test.log test        Videos
count.c~ makefile~    Public      test.c      VirtualBox VMs
data     makefile~   _region_.log test.c~     wait
Desktop  Matlab801   _region_.tex test.cpp    wait.c
dict     missfont.log scr.txt      test.cpp~   wait.c~

tinysHELL$
Waiting a process ...    echo
'end 'end of of batchfile batchfile
tinysHELL$
```

Hình 4: Thực hiện theo lô

### 1.4 Danh sách tiến trình đang thực hiện và hủy bỏ tiến trình

tinysHELL chưa được xây dựng chức năng xem danh sách tiến trình đang thực hiện cũng như hủy bỏ tiến trình. Nhưng có thể dễ dàng sử dụng các chức năng trên nhờ các chương trình của hệ điều hành bao gồm ps và kill. Lệnh ps sẽ hiển thị các chương trình đang chạy, và lệnh kill với tham số PID của tiến trình sẽ hủy bỏ tiến trình đó. Xem ví dụ sau:

```
thongbkvn@EliteBook ~  
tinysHELL$ ./wait&  
Background process haved created      ./wait  
tinysHELL$ ps  
Waiting a process ...      ps  
tinysHELL$ PID TTY          TIME CMD  
20116 pts/9    00:00:00 bash  
20174 pts/9    00:00:00 tinysHELL  
20264 pts/9    00:00:00 wait  
20265 pts/9    00:00:00 ps  
kill 20264  
Waiting a process ...      kill  
tinysHELL$ ps  
Waiting a process ...      ps  
tinysHELL$ PID TTY          TIME CMD  
20116 pts/9    00:00:00 bash  
20174 pts/9    00:00:00 tinysHELL  
20266 pts/9    00:00:00 kill <defunct>  
20267 pts/9    00:00:00 ps
```

Hình 5: Hủy bỏ tiến trình

## 2 Hạn chế của chương trình

Chương trình tinysHELL này rất đơn giản nên có rất nhiều những hạn chế. Một số những hạn chế cơ bản:

- Không thể ngắt chương trình đang thực hiện.
- Chưa có giao diện hướng dẫn người dùng.
- Không thể thực hiện theo kịch bản (script), chỉ thực hiện được theo lô.
- Là chương trình shell nhưng lại ... phải gọi shell khác lên để thực hiện.
- Hiện tượng tiến trình ma (zombie process).

Do vấn đề về thời gian và năng lực của người viết nên chưa thể thực hiện được các chức năng trên.

## 3 Mã nguồn

```
#include <signal.h>  
#include <string.h>  
#include <sys/types.h>  
#include <sys/wait.h>  
#include <stdio.h>  
#include <stdlib.h>  
#include <wait.h>  
#include <string.h>  
#include <unistd.h>  
  
void strstd(char *tmp)  
{  
    int j=0,i;  
    for (i=0;i<strlen(tmp);i++)  
    {  
        if (tmp[i] != ' ' || (tmp[i] == ' ' && tmp[i-1] != ' ' && i>0))
```

```

        {
            tmp[j]=tmp[i];
            j++;
        }
    }
    if (tmp[j-1] == ' ' || tmp[i] == '\n') tmp[j-1] = 0;
    else tmp[j]=0;
}

char **convert(char *cmd)
{
    char * tmp=cmd;
    char **p=(char **) malloc(30*sizeof(char*));
    int i,t=0,length=strlen(cmd);
    for (i=0;i<=length;i++)
    {
        if (cmd[i]==' ')
        {
            cmd[i]=0;
            p[t++]=tmp;
            tmp=cmd+i+1;
        }
        if (cmd[i]==0)
        {
            p[t++]=tmp;
            p[t]=NULL;
        }
    }
    return p;
}

```

```

pid_t spawn(char *cmd)
{
    char **arglist;
    int background=0,status=0;
    pid_t childPid;
    if (cmd[strlen(cmd)-1]=='&')
    {
        background=1;
        cmd[strlen(cmd)-1]=0;
    }
    arglist=convert(cmd);
    childPid=fork();
    if (childPid == 0) //Child process
    {
        execvp(arglist[0],arglist);
        //exec return only if an error occurs.
        printf("An error occurs\n");
        abort();
    }
    else //main process
    {
        if (background==1)

```

```

        {
            printf("\nBackground process haved created \t %s \n",cmd);
        }
        else
        {
            printf("\nWaiting a process ... \t %s \n",cmd);
            wait(&status);
        }
        return childPid;
    }
}

void batch(char* script)
{
    FILE *fp;
    static char tmp[200];
    fp=fopen(script,"r");
    while (!feof(fp))
    {
        printf("\ntinyshell$ ");
        fgets(tmp,200,fp);
        strstd(tmp);
        tmp[strlen(tmp)-1]=0;
        if (strcmp(tmp,""))
            spawn(tmp);
    }
    fclose(fp);
}

char* isbatchscr(char *cmd)
{
    static char tmp[200];
    int i=0;
    while (cmd[i]!=' ')
    {
        tmp[i]=cmd[i];
        i++;
    }
    tmp[i]=0;
    if (!strcmp(tmp,"tinyshell"))
    {
        bzero(tmp,200);
        for (i=10;i<strlen(cmd);i++)
            tmp[i-10]=cmd[i];
        return tmp;
    }
    return NULL;
}

int main(void)
{
    int t=0;
    printf("-----|\n");
    printf(" |          TINY SHELL          |\n");
    printf(" |        Pham Van Thong        |\n");
    printf("-----|\n");

```

```

char *program, **arglist, cmd[200];
pid_t childPid;
while (1)
{
    printf("tinysHELL$ ");
    fgets(cmd,200,stdin);
    cmd[strlen(cmd)-1]=0;
    strstd(cmd);
    if (isbatchscr(cmd))
        batch(isbatchscr(cmd));
    else
        if (strcmp(cmd,""))
            spawn(cmd);
}
}

```