

TINY-SHELL

BÁO CÁO BÀI TẬP LỚN NGUYÊN LÝ HỆ ĐIỀU HÀNH

Mã lớp: 108499

Giảng viên: TS. Phạm Đăng Hải

Thực hiện:

Ngô Việt Hoàng. mssv 20173142

Vũ Hồng Phúc. mssv 20173305



Đại học Bách Khoa Hà Nội
Viện Công nghệ thông tin và Truyền thông

LỜI MỞ ĐẦU

Trong một thế giới công nghệ đang hằng ngày thay đổi, việc học tập kiến thức nền tảng đối với sinh viên kỹ thuật, đặc biệt là sinh viên công nghệ thông tin (CNTT) là vô cùng quan trọng. Bởi lẽ bất cứ công nghệ hiện đại nào cũng được xây dựng nên từ những kiến thức nền tảng cốt lõi, việc học tập những kiến thức này chính là chìa khoá giúp sinh viên CNTT nắm bắt bất cứ công nghệ mới nào một cách nhanh chóng, đầy đủ, rõ ràng về bản chất nhất.

Một trong những kiến thức nền tảng quan trọng nhất mà sinh viên CNTT được học ngay từ những năm đầu đại học là kiến thức về Hệ điều hành. Các kiến thức được học trong môn này về quản lý tiến trình, quản lý bộ nhớ, quản lý hệ thống files... giúp sinh viên hiểu rõ cách thức tổ chức hoạt động của hệ điều hành máy tính.

Hiểu được tầm quan trọng của những kiến thức về Hệ điều hành nói chung và kiến thức về quản lý tiến trình nói riêng, chúng em quyết định thực hiện project Tiny-shell. Project được làm với mục đích tìm hiểu cách cài đặt và hoạt động của một shell đơn giản, cũng như hiểu về một số API quản lý tiến trình do Windows cung cấp.

Về phân chia công việc trong quá trình thực hiện dự án:

- Nhóm cùng thảo luận để đưa ra mô hình thiết kế tổng quát.
- Ngô Việt Hoàng: Xây dựng các chức năng directory, help, đọc file *.bat, các chức năng liên quan đến process.
- Vũ Hồng Phúc: Xây dựng phần thân chương trình (đọc, xử lý tham số đầu vào), các chức năng date/time, path/addpath, kiểm thử.

Toàn bộ nội dung dự án được tóm tắt trong báo cáo này. Ngoài phần mở đầu, mục lục và tài liệu tham khảo, báo cáo gồm 3 phần chính:

Chương 1: Tổng quan dự án, bao gồm một số khái niệm được sử dụng trong báo cáo, mục đích, công nghệ và sơ bộ về sản phẩm.

Chương 2: Cài đặt và sử dụng sản phẩm, bao gồm hướng dẫn sử dụng chi tiết về các chức năng của sản phẩm.

Chương 3: Cách thức xây dựng chương trình, đề cập chi tiết về quá trình phân tích, thiết kế các chức năng của Tiny-shell.

Do là lần đầu tiên thực hiện một dự án thực tế về Hệ điều hành, project này không tránh khỏi những thiếu sót, rất mong nhận được những nhận xét, góp ý của thầy để Tiny-shell và các sản phẩm sau này của chúng em được hoàn thiện hơn.

MỤC LỤC

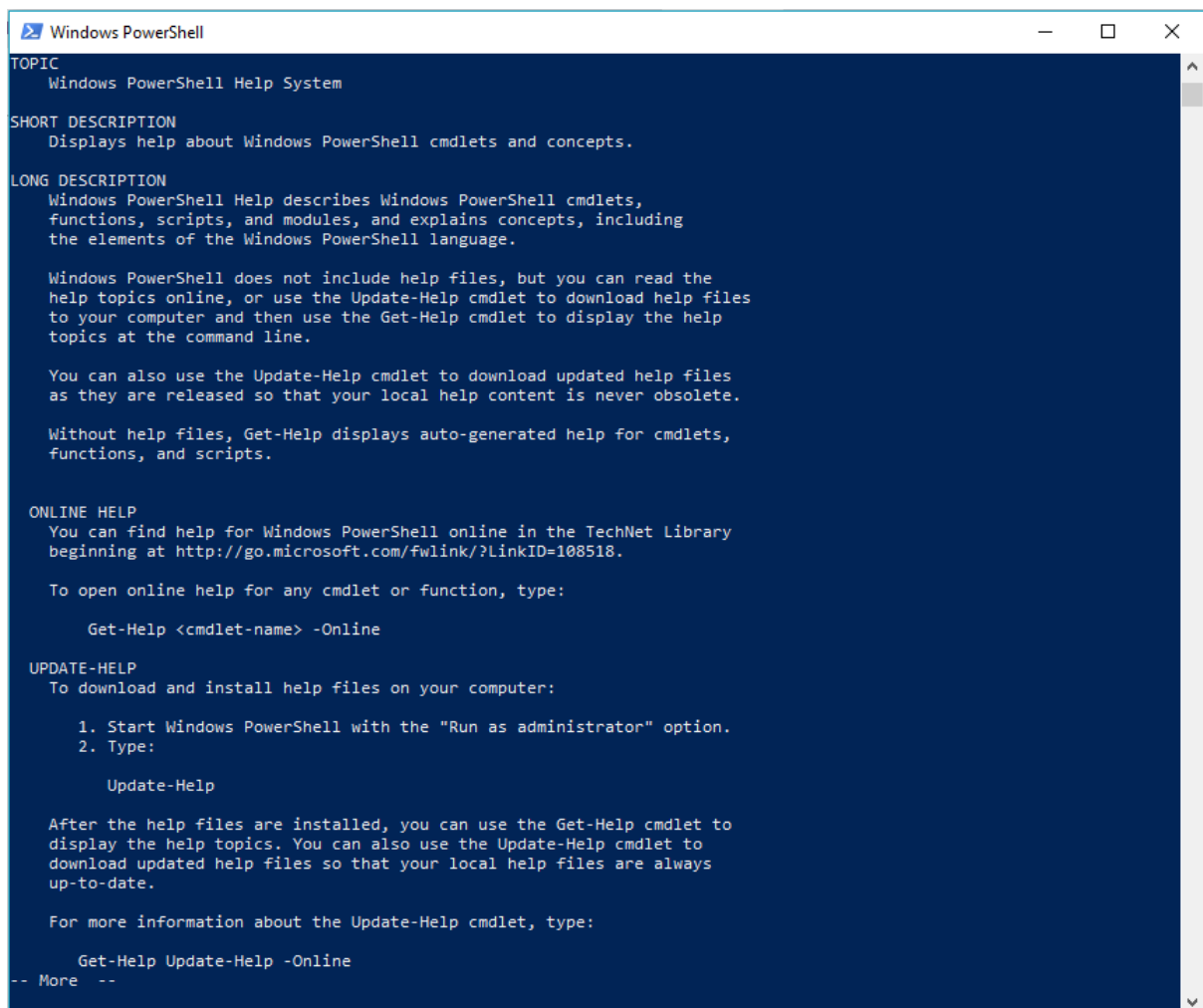
LỜI MỞ ĐẦU	1
MỤC LỤC	2
CHƯƠNG 1. TỔNG QUAN VỀ DỰ ÁN.....	3
1.1 Một số khái niệm.....	3
1.1.1 Shell	3
1.1.2 Tiến trình (Process)	4
1.1.3 API và Windows API	4
1.1.4 Biến môi trường (Environment variable)	4
1.1.5 Tham số dòng lệnh	4
1.2 Tổng quan về dự án.....	5
1.2.1 Mục đích	5
1.2.2 Công nghệ sử dụng	5
1.2.3 Sản phẩm thực hiện được	5
CHƯƠNG 2. CÀI ĐẶT VÀ SỬ DỤNG SẢN PHẨM	6
2.1 Cài đặt tiny-shell	6
2.2 Chi tiết về các chức năng của tiny-shell	6
2.2.1 Help	6
2.2.2 Exit	6
2.2.3 Clear Screen.....	7
2.2.4 Directory	7
2.2.5 Date/time	8
2.2.6 Path	10
2.2.7 Các tác vụ liên quan đến tiến trình.....	11
CHƯƠNG 3. CÁCH THỨC XÂY DỰNG CHƯƠNG TRÌNH	18
3.1 Một số kiểu dữ liệu trong Window mà chương trình sử dụng	18
3.2 Tổng quan về cách thực hiện chương trình.....	21
3.3 Tiến trình và các tác vụ liên quan đến tiến trình.....	22
3.4 Help/Exit/Clear Screen	25
3.5 Directory.....	25
3.6 Date/Time.....	26
3.7 Path/AddPath	26
TÀI LIỆU THAM KHẢO	28

CHƯƠNG 1. TỔNG QUAN VỀ DỰ ÁN

1.1 Một số khái niệm

1.1.1 Shell

Shell là một giao diện diện người dùng cho phép truy cập vào các dịch vụ của hệ điều hành. Chương trình này được gọi là shell vì nó tạo thành một lớp bao quanh nhân (kernel) của hệ điều hành. Trong đa số các trường hợp, người sử dụng tương tác với shell thông qua giao diện dòng lệnh (*command-line interface – CLI*) và giao diện đồ họa (*graphical user interface – GUI*), ngoài ra người dùng cũng có thể thao tác với giao diện giọng nói (*voice user interface*) và *text-based user interface – TUI*.



Hình 1. Windows PowerShell, một CLI shell do hệ điều hành Window cung cấp

CLI shells, đúng như tên gọi của nó, cho phép người dùng sử dụng các dịch vụ của hệ điều hành thông qua các ký tự được nhập vào từ bàn phím (bao gồm các lệnh, tham số đầu vào, ...) hoặc các đoạn shell script. Một số CLI thông dụng như Command Prompt (CMD), PowerShell trên Windows, Bash Shell trên các hệ điều hành họ Unix. MS-DOS là một hệ điều hành chỉ thuần giao diện dòng lệnh.

1.1.2 Tiến trình (Process)

Có nhiều định nghĩa về tiến trình. Trong cuốn *Operating System Internals and Design Principles*, W.Stallings đã nêu ra ít nhất 4 định nghĩa về tiến trình. Hiểu một cách thông dụng nhất, tiến trình là sự thực hiện chương trình trên máy tính.

1.1.3 API và Windows API

API (*Application Programing Interface* – Giao diện lập trình ứng dụng) là một giao diện được cung cấp bởi hệ điều hành hoặc các ứng dụng cho phép những ứng dụng này giao tiếp với nhau hoặc giao tiếp với hệ điều hành, thực hiện các hoạt động như yêu cầu dịch vụ, yêu cầu tài nguyên hệ thống, trao đổi dữ liệu...

Hệ điều hành Windows cung cấp các hàm API để các chương trình ứng dụng có thể giao tiếp với hệ điều hành, bao gồm phiên bản cho hệ điều hành 32-bit (WIN32 API) và 64-bit (WIN64 API).

1.1.4 Biến môi trường (Environment variable)

Biến môi trường là một đối tượng động (dynamic) trên máy tính. Mỗi biến môi trường bao gồm một cặp name – value, trong đó value có thể thay đổi được. Biến môi trường được sử dụng bởi một hoặc nhiều phần mềm trong hệ điều hành và giúp định hình môi trường mà các ứng dụng sẽ chạy trên hệ điều hành.

Biến môi trường PATH có giá trị là đường dẫn tới thư mục chứa file thực thi (executable file) mà shell sẽ tìm kiếm khi người dùng nhập lệnh tương ứng từ bàn phím.

```
PATHEXT=.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
PROCESSOR_ARCHITECTURE=x86
PROCESSOR_ARCHITECTURE_AMD64=AMD64
PROCESSOR_IDENTIFIER=Intel64 Family 6 Model 142 Stepping 9, GenuineIntel
PROCESSOR_LEVEL=6
PROCESSOR_REVISION=8e09
ProgramData=C:\ProgramData
ProgramFiles=C:\Program Files (x86)
ProgramFiles(x86)=C:\Program Files (x86)
ProgramW6432=C:\Program Files
PROMPT=$P$G
PSModulePath=C:\Program Files\WindowsPowerShell\Modules;C:\WINDOWS\system32\WindowsPowerShell\v1.0\Modules
PUBLIC=C:\Users\Public
PythonPath=C:\Users\MyPC\AppData\Local\Programs\Python\Python37-32\python.exe
SESSIONNAME=Console
SynaProgDir=Synaptics\SynTP
```

Hình 2. Một số tham số dòng lệnh trên Windows, mỗi tham số được cho dưới dạng name = value.

1.1.5 Tham số dòng lệnh

Tham số dòng lệnh là các tham số tùy ý mà người dùng cung cấp cho chương trình dưới dạng chuỗi khi thực thi. Các tham số này được truyền vào thông qua hệ điều hành và được chương trình sử dụng như là input của chương trình. Hai giá trị được sử dụng để quản lý các tham số dòng lệnh là: số tham số dòng lệnh (*argument count* – *argc*) và danh sách các tham số dòng lệnh (*argument vector* – *argv[]*).

Xét câu lệnh sau:

settime 15:03:20

Câu lệnh trên có 2 tham số dòng lệnh là *settime* và *15:03:20*, ngăn cách nhau bởi dấu ‘ ‘ (space), tương ứng với $argc = 2$, $argv[] = \{settime, 15:03:20\}$.

1.2 Tổng quan về dự án

1.2.1 Mục đích

Xây dựng một CLI shell đơn giản theo mô hình Command Prompt do hệ điều hành Windows cung cấp.

1.2.2 Công nghệ sử dụng

Dự án sử dụng ngôn ngữ C để xây dựng ứng dụng. Các chức năng của Shell được xây dựng chủ yếu dựa trên các hàm API được Windows cung cấp sẵn.

1.2.3 Sản phẩm thực hiện được

Sản phẩm có thể thực hiện một số chức năng sau:

- Hiểu và thực hiện một số lệnh như **dir**, **date/time**, **exit**, **help**, **path**, **addpath**, ...
- Tạo tiến trình **background** và **foreground**.
- Thực hiện một số câu lệnh quản lý tiến trình: **list**, **kill**, **stop**, **resume**,...
- Đọc file ***.bat**.

Chi tiết về các chức năng của tiny-shell sẽ được trình bày trong chương 2.

Mã nguồn của toàn bộ dự án trên github tại địa chỉ:

<https://github.com/deserteagle735/tiny-shell>

CHƯƠNG 2. CÀI ĐẶT VÀ SỬ DỤNG SẢN PHẨM

2.1 Cài đặt tiny-shell

Sản phẩm được build ra file *.exe và chạy trực tiếp trên máy

2.2 Chi tiết về các chức năng của tiny-shell

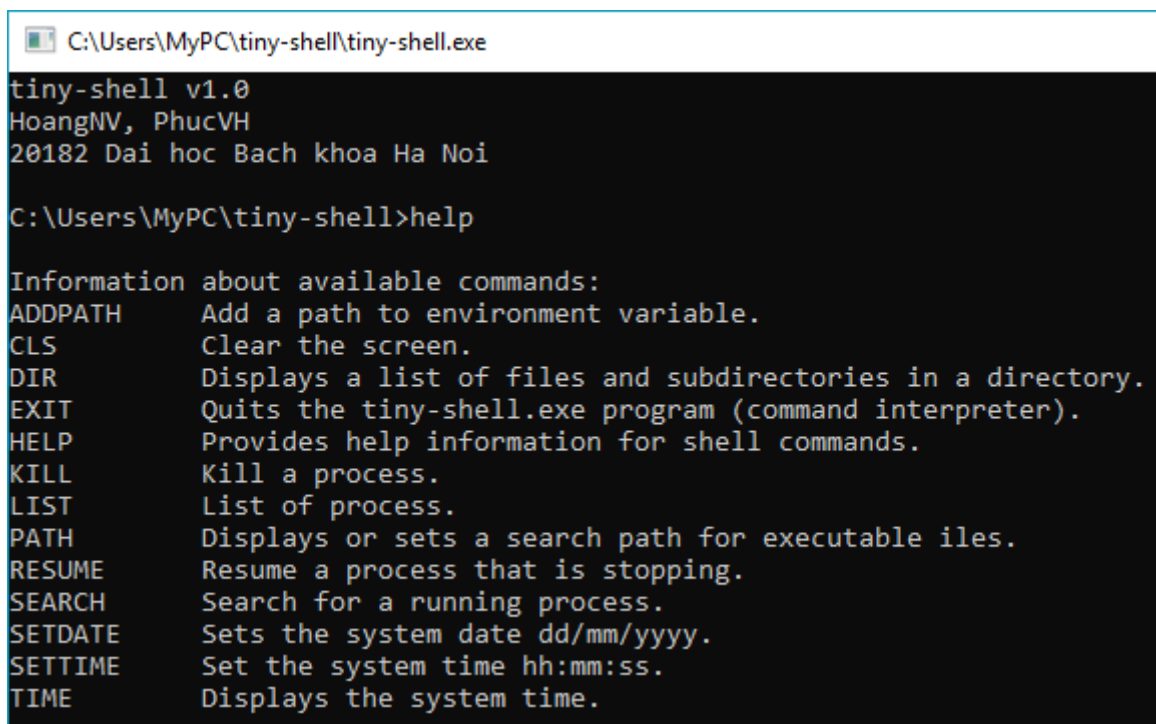
2.2.1 Help

Cú pháp:

help

Tham số: không có tham số truyền vào ($argc = 1, argv[] = \{help\}$)

Chức năng: Cung cấp thông tin về các câu lệnh được hỗ trợ bởi shell.



```
C:\Users\MyPC\tiny-shell\tiny-shell.exe
tiny-shell v1.0
HoangNV, PhucVH
20182 Dai hoc Bach khoa Ha Noi
C:\Users\MyPC\tiny-shell>help

Information about available commands:
ADDPATH      Add a path to environment variable.
CLS          Clear the screen.
DIR          Displays a list of files and subdirectories in a directory.
EXIT         Quits the tiny-shell.exe program (command interpreter).
HELP         Provides help information for shell commands.
KILL         Kill a process.
LIST         List of process.
PATH         Displays or sets a search path for executable files.
RESUME       Resume a process that is stopping.
SEARCH       Search for a running process.
SETDATE      Sets the system date dd/mm/yyyy.
SETTIME      Set the system time hh:mm:ss.
TIME         Displays the system time.
```

Hình 3. Kết quả thực hiện lệnh help

2.2.2 Exit

Cú pháp:

exit

Tham số: không có tham số truyền vào ($argc = 1, argv[] = \{exit\}$)

Chức năng: thoát khỏi chương trình shell đang chạy.

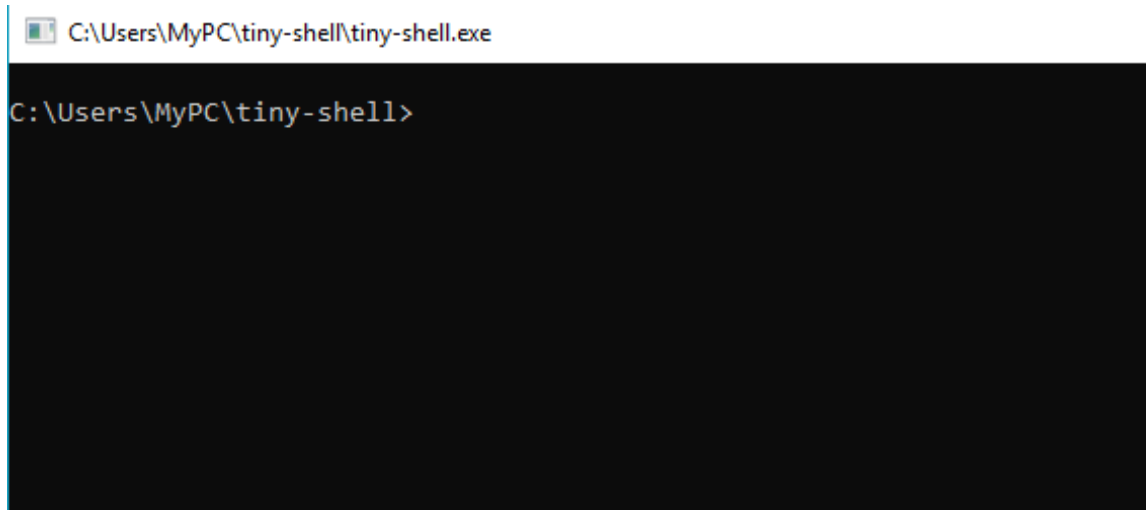
2.2.3 Clear Screen

Cú pháp:

cls

Tham số: không có tham số truyền vào ($argc = 1, argv[] = \{cls\}$)

Chức năng: xoá màn hình kết quả thực hiện của các câu lệnh trước.



Hình 4. Kết quả thực hiện lệnh *cls*

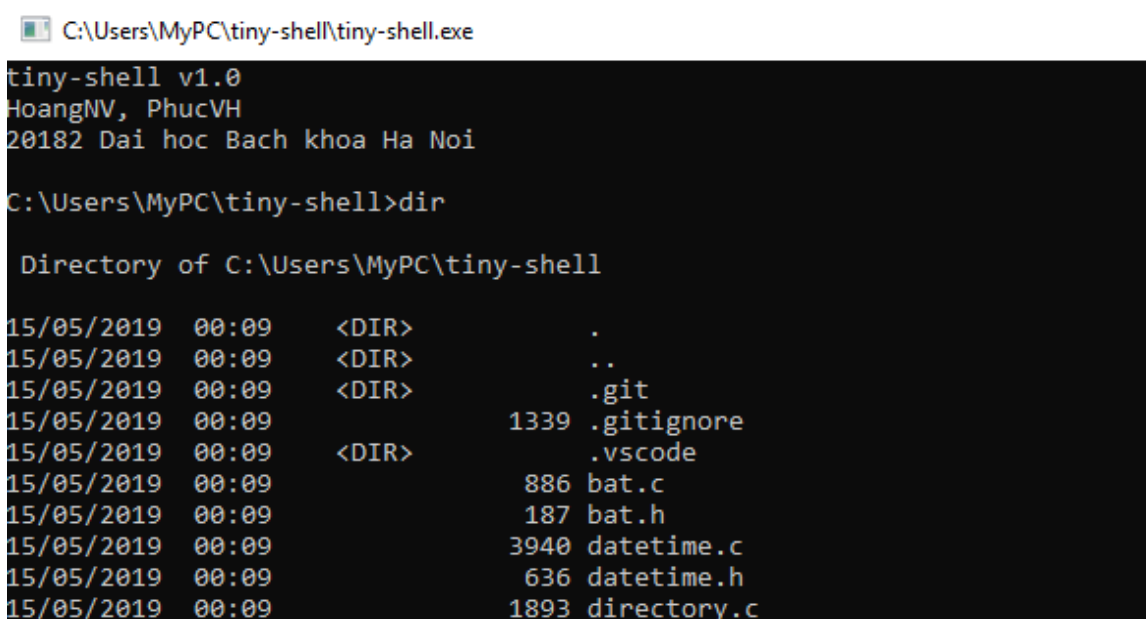
2.2.4 Directory

Cú pháp:

dir

Tham số: không có tham số truyền vào ($argc = 1, argv[] = \{help\}$)

Chức năng: Liệt kê các tệp và thư mục con của thư mục chứa shell hiện tại.



Hình 5. Kết quả thực hiện lệnh *dir*

2.2.5 Date/time

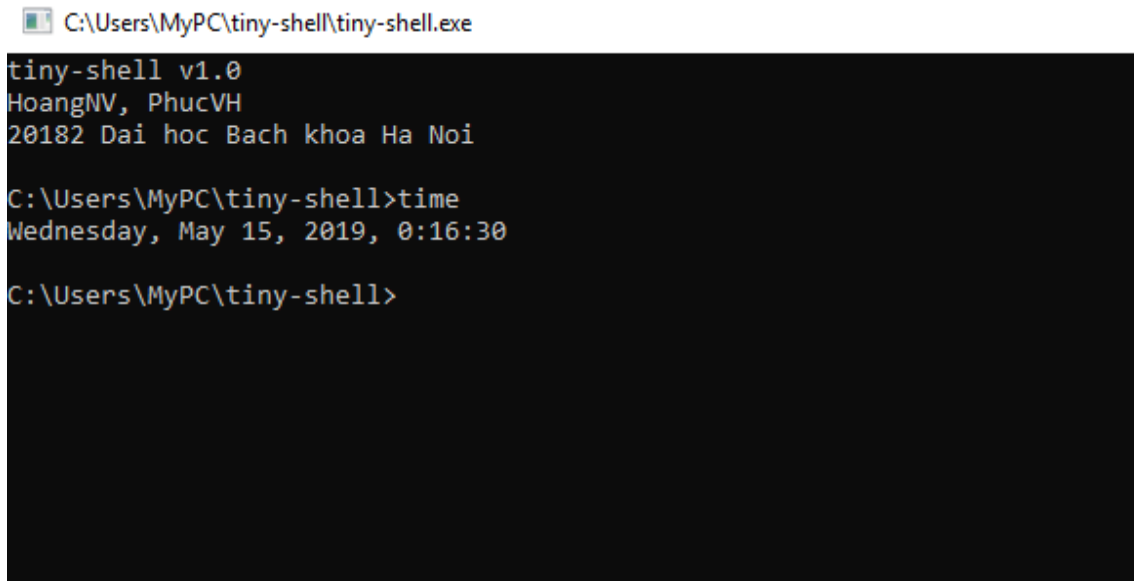
2.2.5.1 Time

Cú pháp:

time

Tham số: không có tham số truyền vào ($argc = 1, argv[] = \{help\}$)

Chức năng: In ra thời gian hiện tại của hệ thống (Theo thứ tự bao gồm Thứ, Tháng Ngày, Năm, Giờ:Phút:Giây)



Hình 6. Kết quả thực hiện lệnh time

2.2.5.2 Set time

Cú pháp:

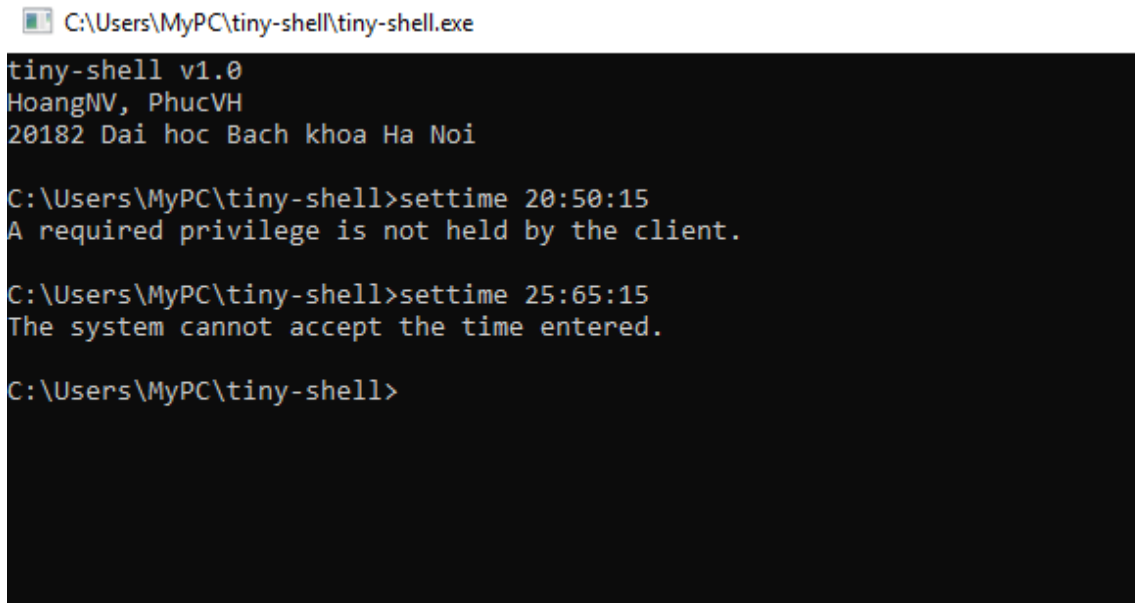
 $settime <hh:mm:ss>$

Tham số: lệnh *settime* nhận tham số là chuỗi *hh:mm:ss* (giờ:phút:giây) (*argc* = 2, *argv*[] = {*settime*, *hh:mm:ss*}).

Chức năng: Thay đổi thời gian (giờ, phút, giây) của hệ thống từ giá trị hiện tại thành giá trị do người dùng cung cấp.

Lưu ý:

- Giá trị giờ, phút, giây phải được nhập chính xác (mỗi giá trị gồm 1 hoặc 2 chữ số, $hh < 24$, $mm < 60$, $ss < 60$) vì lệnh *settime* kiểm tra đầu vào chặt chẽ, nếu đầu vào sai, chương trình sẽ đưa ra thông báo “*The system cannot accept the time entered.*”
- Cần chạy chương trình shell với quyền administrator để thực hiện lệnh *settime*, do lệnh này làm thay đổi thời gian hệ thống. Nếu không, chương trình sẽ đưa ra thông báo: “*A required privilege is not held by the client.*”



```

C:\Users\MyPC\tiny-shell\tiny-shell.exe
tiny-shell v1.0
HoangNV, PhucVH
20182 Dai hoc Bach khoa Ha Noi

C:\Users\MyPC\tiny-shell>settime 20:50:15
A required privilege is not held by the client.

C:\Users\MyPC\tiny-shell>settime 25:65:15
The system cannot accept the time entered.

C:\Users\MyPC\tiny-shell>

```

Hình 7. Kết quả thực hiện lệnh `settime` với thời gian hợp lệ nhưng không chạy với quyền administrator (trên) và với thời gian không hợp lệ (dưới).

2.2.5.3 Set date

Cú pháp:

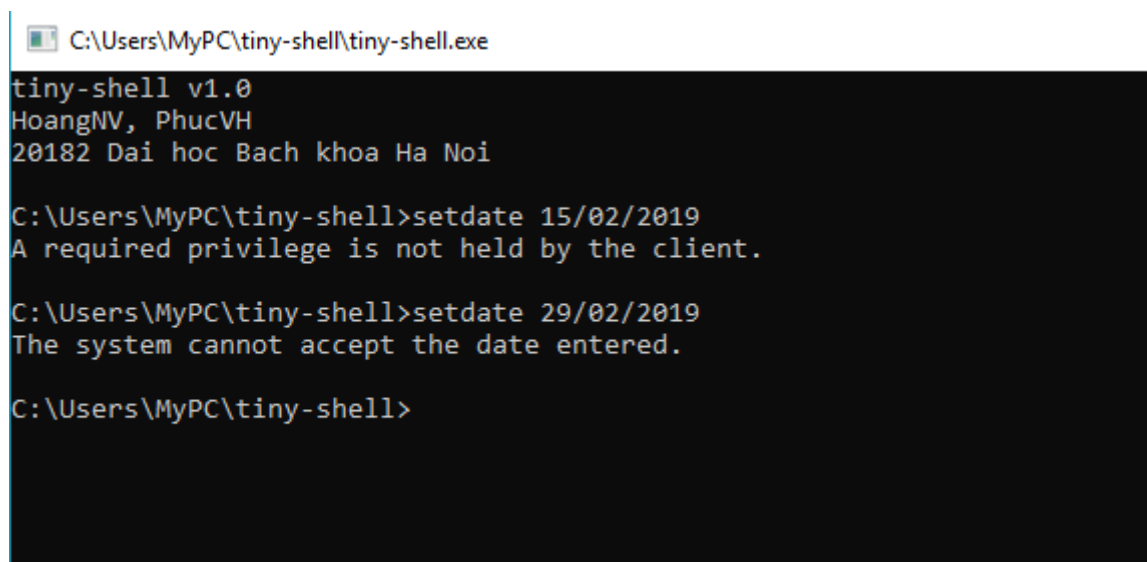
setdate <dd/mm/yy>

Tham số: lệnh `setdate` nhận tham số là chuỗi `dd/mm/yy` (ngày/tháng/năm) ($argc = 2$, $argv[] = \{setdate, dd/mm/yy\}$).

Chức năng: Thay đổi thời gian (ngày, tháng, năm) của hệ thống từ giá trị hiện tại thành giá trị do người dùng cung cấp.

Lưu ý:

- Giá trị ngày, tháng, năm phải được nhập chính xác (mỗi giá trị gồm 1 hoặc 2 chữ số đối với ngày và tháng, giá trị ngày phải phù hợp với tháng và năm, $1601 < yy < 30827$) vì lệnh `settime` kiểm tra đầu vào chặt chẽ, nếu đầu vào sai, chương trình sẽ đưa ra thông báo *"The system cannot accept the date entered."*
- Cần chạy chương trình shell với quyền administrator để thực hiện lệnh `setdate`, do lệnh này làm thay đổi thời gian hệ thống. Nếu không, chương trình sẽ đưa ra thông báo: *"A required privilege is not held by the client."*



```

C:\Users\MyPC\tiny-shell\tiny-shell.exe
tiny-shell v1.0
HoangNV, PhucVH
20182 Dai hoc Bach khoa Ha Noi

C:\Users\MyPC\tiny-shell>setdate 15/02/2019
A required privilege is not held by the client.

C:\Users\MyPC\tiny-shell>setdate 29/02/2019
The system cannot accept the date entered.

C:\Users\MyPC\tiny-shell>

```

Hình 8. Kết quả thực hiện lệnh `setdate` với ngày tháng năm hợp lệ nhưng không chạy với quyền administrator (trên) và với thời gian không hợp lệ (dưới).

2.2.6 Path

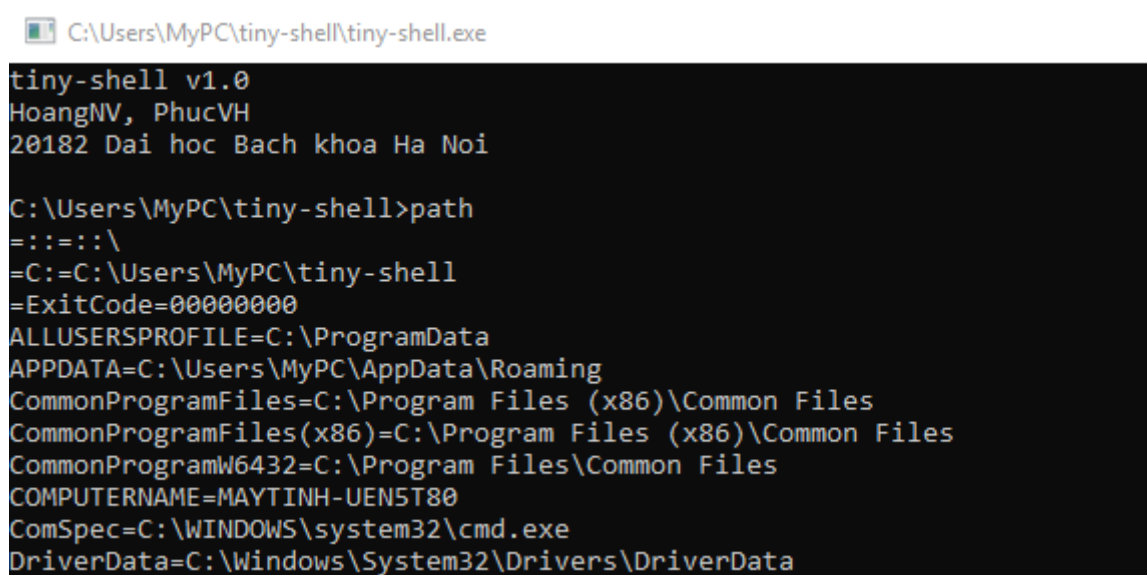
2.2.6.1 Path

Cú pháp:

path

Tham số: không có tham số truyền vào ($argc = 1$, $argv[] = \{help\}$)

Chức năng: nếu thành công, chương trình sẽ in ra danh sách các biến môi trường hiện tại, nếu không in ra thông báo “*Cannot find the pointer to the environment block of the current process*”.



```

C:\Users\MyPC\tiny-shell\tiny-shell.exe
tiny-shell v1.0
HoangNV, PhucVH
20182 Dai hoc Bach khoa Ha Noi

C:\Users\MyPC\tiny-shell>path
=::=::\n
=C:=C:\Users\MyPC\tiny-shell\n
=ExitCode=00000000\n
ALLUSERSPROFILE=C:\ProgramData\n
APPDATA=C:\Users\MyPC\AppData\Roaming\n
CommonProgramFiles=C:\Program Files (x86)\Common Files\n
CommonProgramFiles(x86)=C:\Program Files (x86)\Common Files\n
CommonProgramW6432=C:\Program Files\Common Files\n
COMPUTERNAME=MAYTINH-UEN5T80\n
ComSpec=C:\WINDOWS\system32\cmd.exe\n
DriverData=C:\Windows\System32\Drivers\DriverData\n

```

Hình 9. Kết quả thực thi lệnh `path`

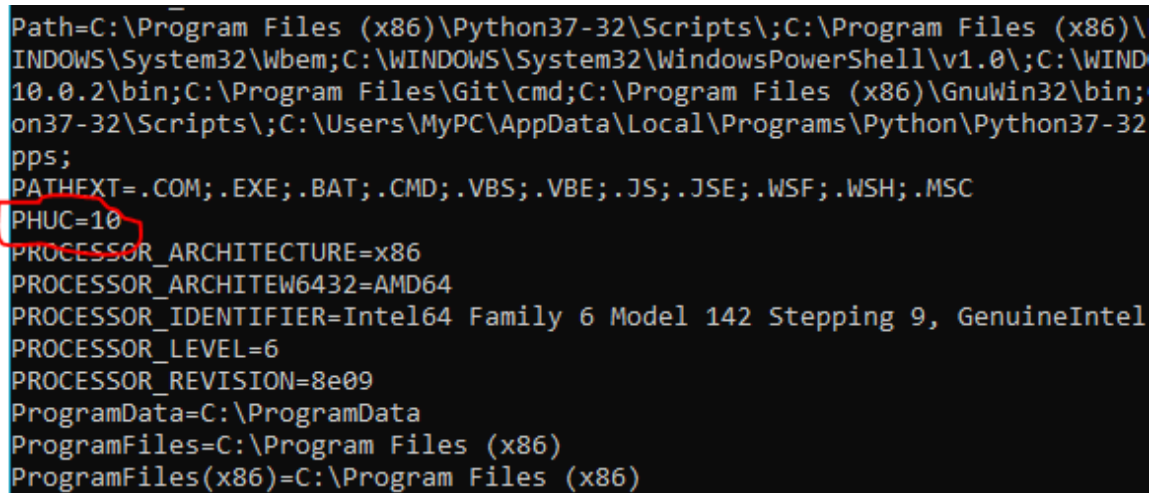
2.2.6.2 Add Path

Cú pháp:

addpath <name> <value>

Tham số: lệnh addpath nhận 2 tham số đầu vào là tên biến môi trường (*name*) và giá trị của nó (*value*) (*argc* = 3, *argv*[] = {*addpath*, *name*, *value*})

Chức năng: nếu thành công, biến môi trường *name* với giá trị *value* sẽ được thêm vào phần user environment variable của hệ điều hành và danh sách biến môi trường mới được in ra, ngược lại nếu thực hiện không thành công, chương trình đưa ra thông báo: "SetEnvironmentVariable failed <lỗi gặp phải>"



```
Path=C:\Program Files (x86)\Python37-32\Scripts\;C:\Program Files (x86)\
WINDOWS\System32\Wbem;C:\WINDOWS\System32\WindowsPowerShell\v1.0\;C:\WIND
10.0.2\bin;C:\Program Files\Git\cmd;C:\Program Files (x86)\GnuWin32\bin;
on37-32\Scripts\;C:\Users\MyPC\AppData\Local\Programs\Python\Python37-32
pps;
PATHEXT=.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
PHUC=10
PROCESSOR_ARCHITECTURE=x86
PROCESSOR_ARCHITECTUREAMD64=AMD64
PROCESSOR_IDENTIFIER=Intel64 Family 6 Model 142 Stepping 9, GenuineIntel
PROCESSOR_LEVEL=6
PROCESSOR_REVISION=8e09
ProgramData=C:\ProgramData
ProgramFiles=C:\Program Files (x86)
ProgramFiles(x86)=C:\Program Files (x86)
```

Hình 10. Kết quả thu được khi chạy câu lệnh *addpath PHUC = 10*.

2.2.7 Các tác vụ liên quan đến tiến trình

Ngoại trừ lệnh *list*, các lệnh khác liên quan đến tiến trình đều có thể thực hiện bằng hai cách, hoặc sử dụng tên tiến trình, hoặc sử dụng ID của tiến trình (được cung cấp bởi lệnh *list*)

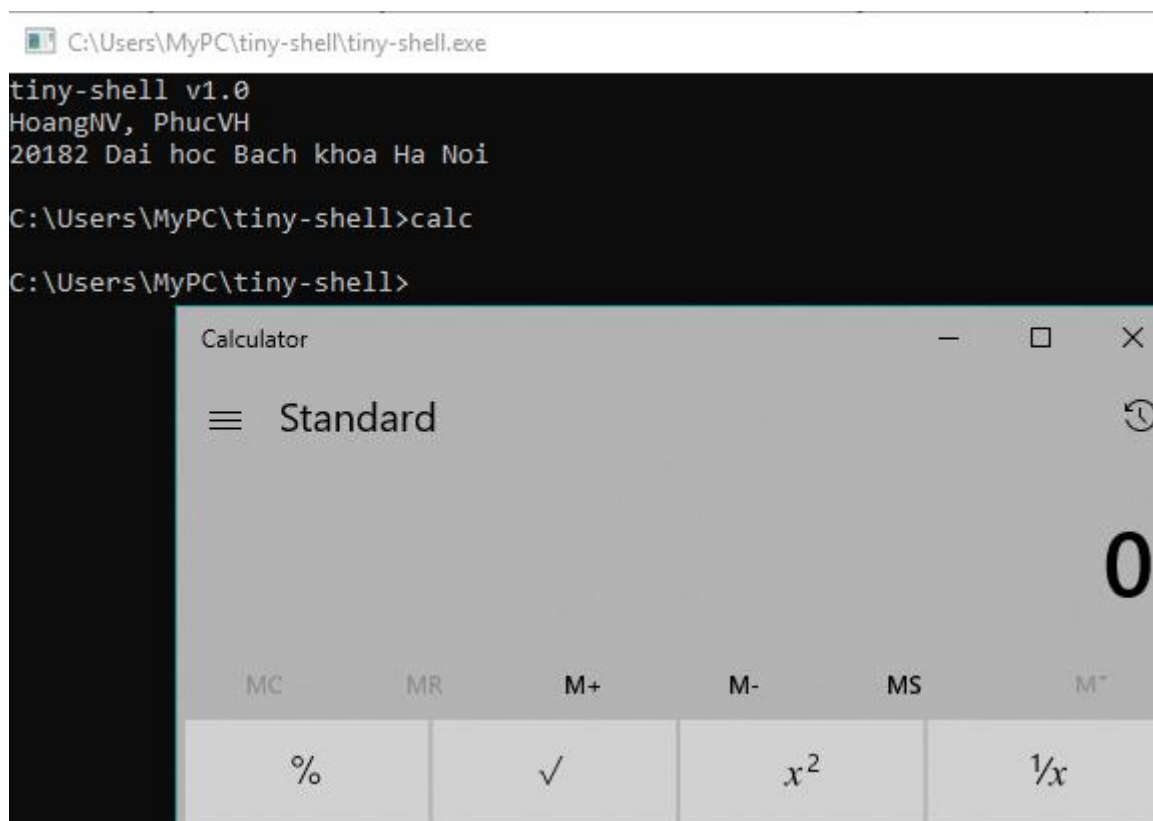
Chương trình *tiny-shell* này không chỉ quản lý các tiến trình được người dùng tạo ra bằng cách dùng chính *tiny-shell*, mà còn tác động được đến các tiến trình mà hệ điều hành đang quản lý.

2.2.7.1 Tiến trình *Foreground* và tiến trình *Background*

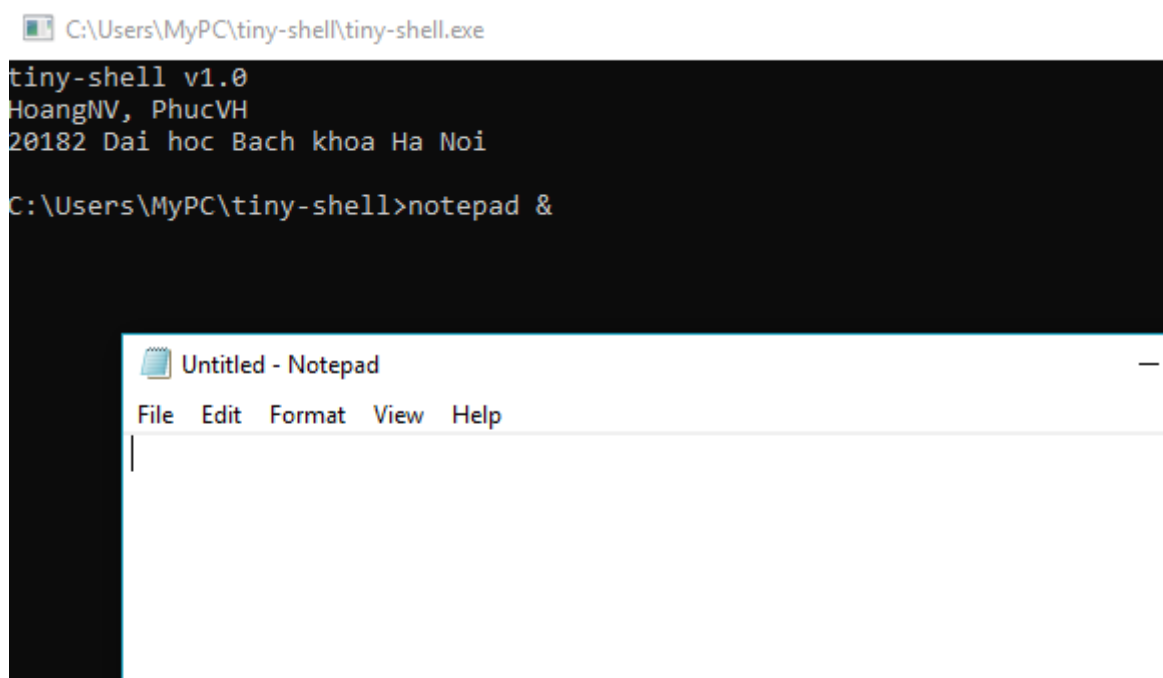
Tiny-shell có thể gọi các chương trình theo biến môi trường như *notepad*, *calculator*,...

Có hai cách gọi tiến trình là tiến trình **background** và tiến trình **foreground**. Để gọi tiến trình **background** chỉ cần nhập tên biến môi trường tương ứng (ví dụ *notepad*, *tasklist*, *calc*,...), còn để gọi tiến trình **foreground** cần thêm dấu *&* vào sau lệnh gọi (ví dụ *notepad &*, *tasklist &*, *calc &*,...).

Để huỷ một tiến trình **foreground** đang chạy ta sử dụng tổ hợp phím **Ctrl + C**.



Hình 11. Kết quả khi gọi một tiến trình Background. Chương trình có thể nhận ngay lệnh tiếp theo.



Hình 12. Kết quả khi gọi một tiến trình Foreground, chương trình đợi tiến trình kết thúc mới nhận lệnh tiếp theo.

2.2.7.2 Đọc và thực hiện file *.bat

Chương trình có thể đọc và thực hiện file *filename.bat* theo cú pháp:

`<filename.bat>`

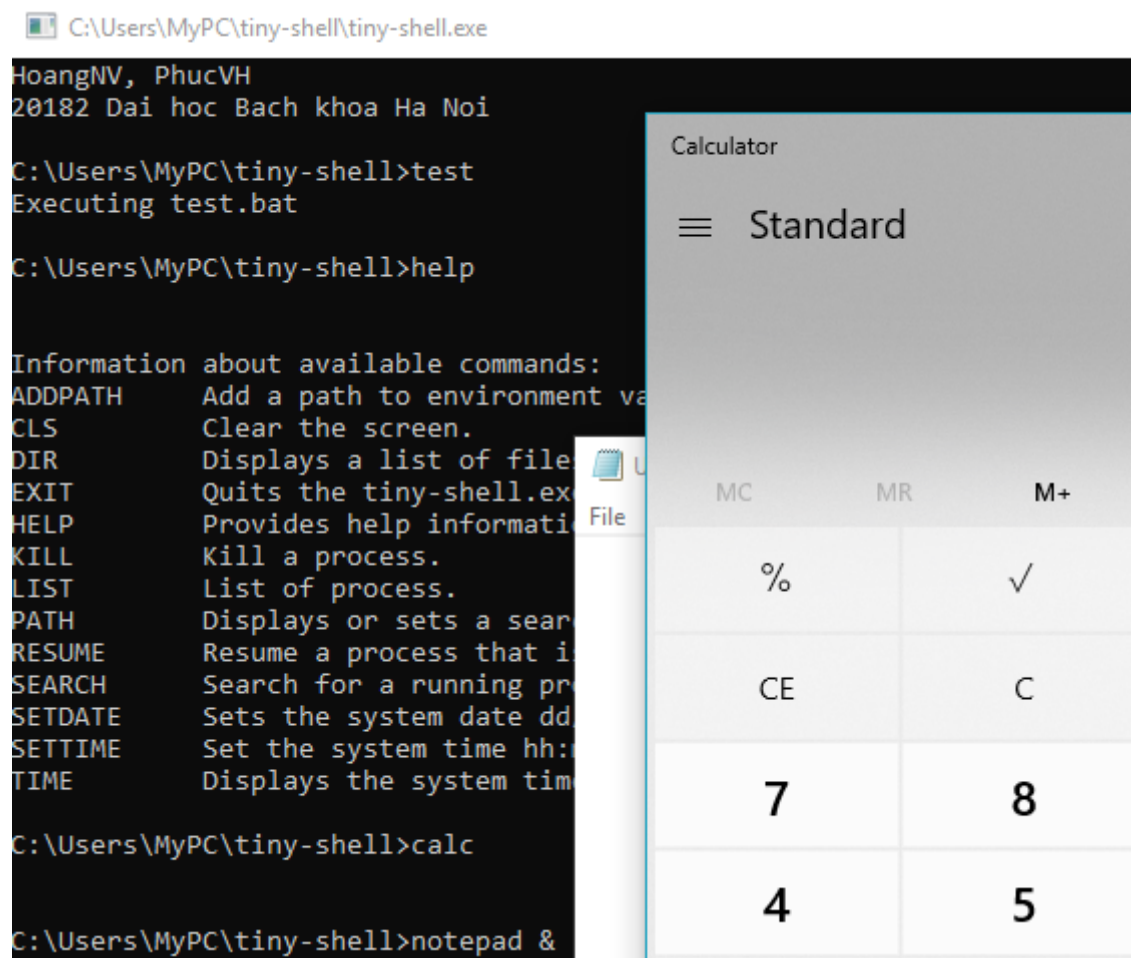
hoặc:

`<filename>`

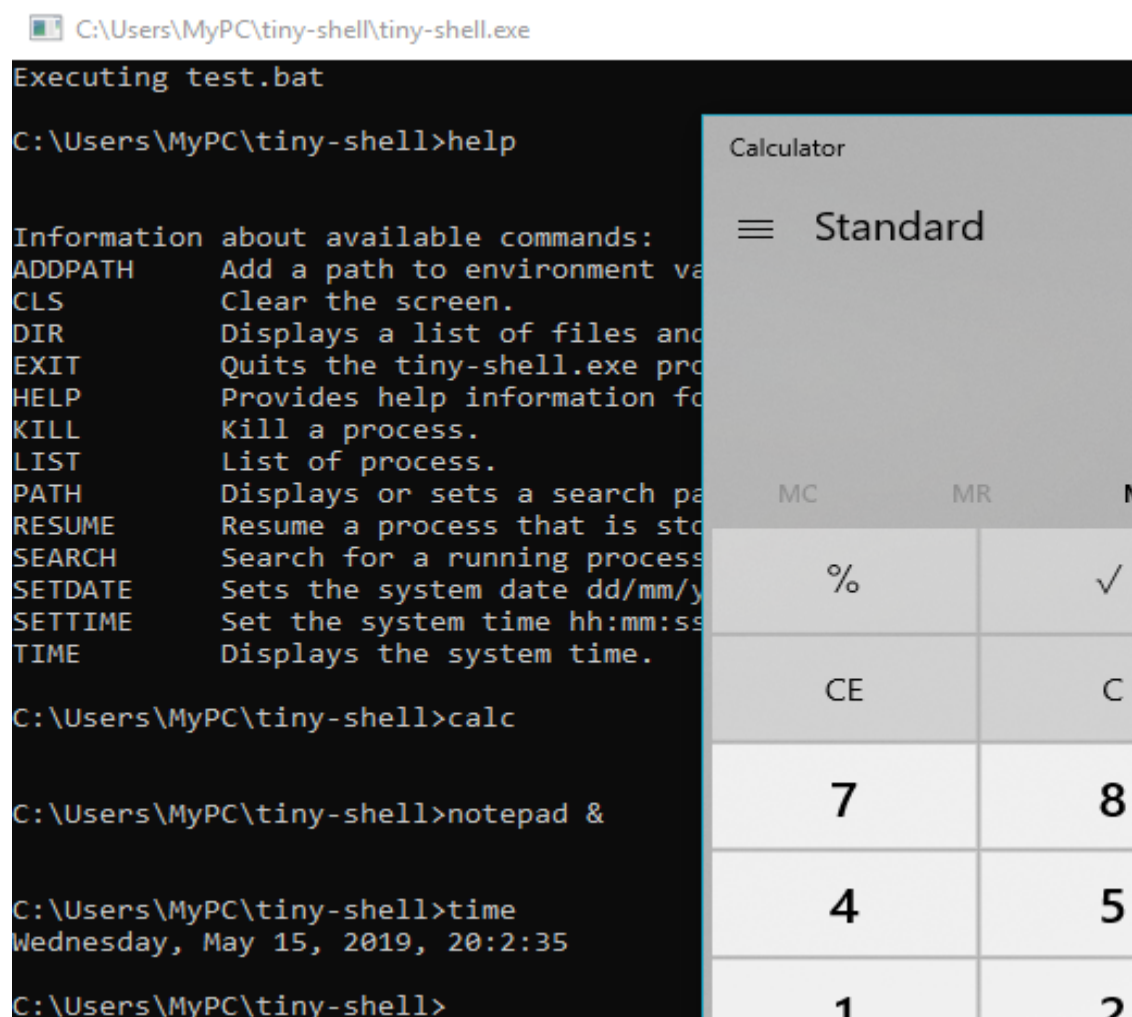
Ví dụ: Thực thi file *.bat có nội dung dưới đây:

```
test - Notepad
File Edit Format View Help
help
calc
notepad &
time
```

Hình 13. Nội dung file *.bat.



Hình 14. Kết quả thực thi file *.bat ở trên.



Hình 15. Kết quả thực thi file *.bat ở trên. Tiến trình notepad đã kết thúc sau khi nhấn tổ hợp phím Ctrl + C.

2.2.7.3 List

Cú pháp:

list

hoặc:

list all

Tham số: không có tham số truyền vào ($argc = 1, argv[] = \{help\}$)

Chức năng:

- Lệnh *list* liệt kê danh sách các tiến trình, không bao gồm các tiến trình của hệ điều hành, với các thông số PROCESS NAME (tên tiến trình), Process ID (Giá trị định danh đặc trưng cho từng tiến trình), Thread count (số luồng của mỗi tiến trình).
- Lệnh *list all* liệt kê danh sách tất cả các tiến trình, bao gồm cả các tiến trình được hệ điều hành bảo vệ với các tham số tương tự lệnh *list*.

```
C:\Users\MyPC\tiny-shell\tiny-shell.exe
tiny-shell v1.0
HoangNV, PhucVH
20182 Dai hoc Bach khoa Ha Noi

C:\Users\MyPC\tiny-shell>list

PROCESS NAME:  SynTPEnh.exe
  Process ID   = 9492
  Thread count = 7
PROCESS NAME:  svchost.exe
  Process ID   = 7488
  Thread count = 11
PROCESS NAME:  sihost.exe
  Process ID   = 4708
  Thread count = 14
PROCESS NAME:  svchost.exe
  Process ID   = 9984
  Thread count = 10
```

Hình 16. Kết quả thực hiện lệnh list, lệnh list all sẽ cho kết quả tương tự.

2.2.7.4 Search

Cú pháp:

search <process name> (tìm kiếm theo tên tiến trình)

hoặc:

```
search pid <process ID>
```

Tham số:

- Lệnh `search <process name>` nhận 1 tham số dòng lệnh là `<process name>` (tên của tiến trình cần tìm kiếm).
- Lệnh `search pid <process ID>` nhận 2 tham số, tham số thứ nhất là `pid` để phân biệt lệnh với lệnh `search <name>`, tham số thứ hai là `<process ID>` (ID của tiến trình cần tìm).

Chức năng:

- Lệnh `search <name>` sẽ tìm **tất cả** các tiến trình có tên `name` trong danh sách tiến trình, đưa ra thông tin bao gồm tên tiến trình, ID của tiến trình và số luồng hiện tại của tiến trình


```
C:\Users\MyPC\tiny-shell\tiny-shell.exe
tiny-shell v1.0
HoangNV, PhucVH
20182 Dai hoc Bach khoa Ha Noi

C:\Users\MyPC\tiny-shell>search tiny-shell.exe

PROCESS NAME:  tiny-shell.exe
  Process ID      = 8928
  Thread count    = 1
PROCESS NAME:  tiny-shell.exe
  Process ID      = 5000
  Thread count    = 1
PROCESS NAME:  tiny-shell.exe
  Process ID      = 5400
  Thread count    = 3
```

Hình 17. Kết quả thực hiện lệnh `search tiny-shell.exe`.

- Lệnh *search pid* *<process ID>* in ra tên, ID và số luồng hiện tại của tiến trình có định danh *process ID*.

```
C:\Users\MyPC\tiny-shell\tiny-shell.exe
tiny-shell v1.0
HoangNV, PhucVH
20182 Dai hoc Bach khoa Ha Noi

C:\Users\MyPC\tiny-shell>search pid 9104

PROCESS NAME:  FoxitReader.exe
  Process ID      = 9104
  Thread count    = 18
```

Hình 18. Kết quả thực hiện lệnh `search pid 9104`.

2.2.7.5 Stop

Cú pháp:

stop *<process name>*

hoặc:

stop pid <process ID>

Tham số: các tham số của lệnh stop tương tự tham số của lệnh *search*.

Chức năng:

- Lệnh `stop <process name>` sẽ tạm dừng tiến trình **đầu tiên** có tên *process name* trong danh sách tiến trình đang chạy. Nếu thành công, chương trình sẽ đưa ra thông báo “*Process Stopped*”. Ngược lại, nếu không tìm thấy tiến trình có tên *process name* trong danh sách tiến trình, shell sẽ đưa ra thông báo “*Process not found*”.
- Lệnh `stop pid <process ID>` sẽ tạm dừng tiến trình có định danh *process ID* trong danh sách tiến trình, cho kết quả tương tự như lệnh `stop <process name>` nếu thành công.

CHƯƠNG 3. CÁCH THỨC XÂY DỰNG CHƯƠNG TRÌNH

3.1 Một số kiểu dữ liệu trong Window mà chương trình sử dụng

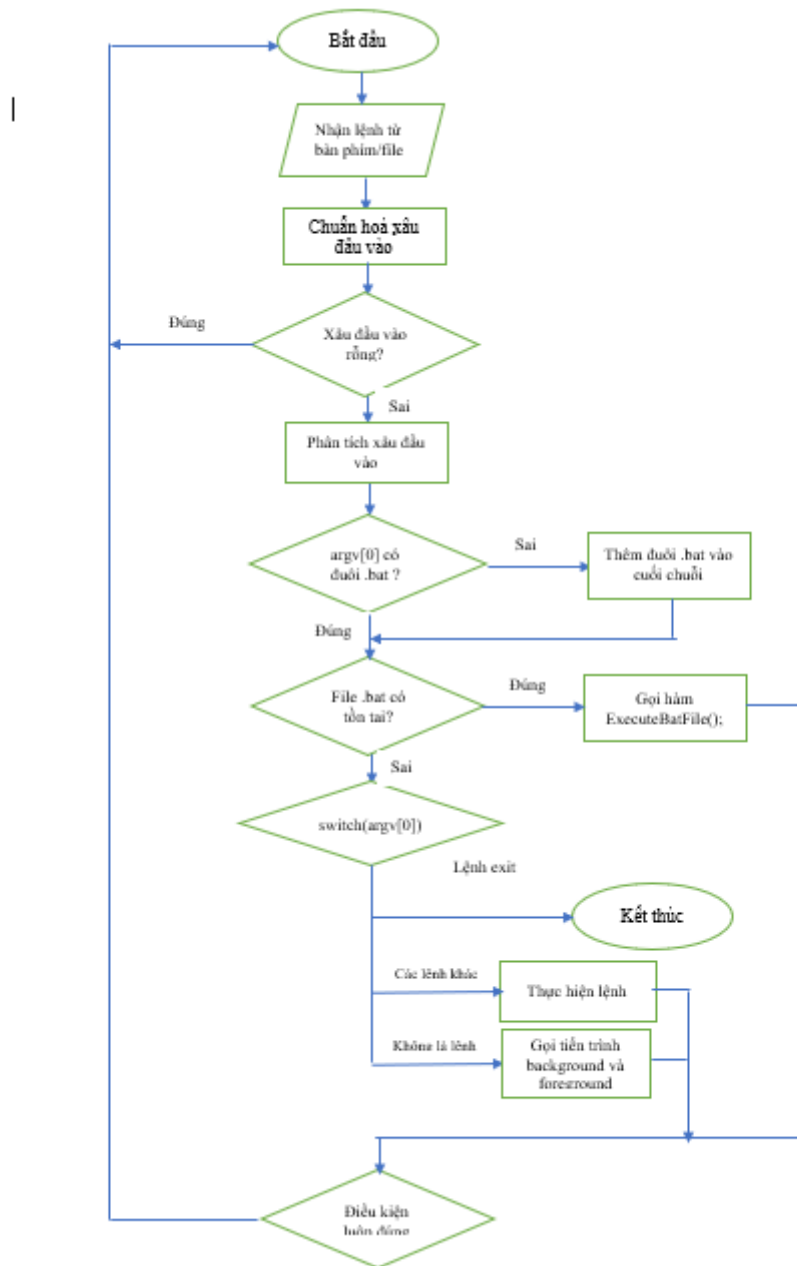
Kiểu dữ liệu	Định nghĩa	Mô tả
DWORD	<code>typedef unsigned long DWORD</code>	Số nguyên không dấu
TCHAR	<code>#ifdef UNICODE typedef wchar_t TCHAR #else typedef unsigned char TCHAR</code>	Kiểu kí tự
LPTSTR	<code>typedef TCHAR *LPTSTR, *LPTCH;</code>	Con trỏ kiểu TCHAR
LPTCH		
HANDLE	<code>typedef void *PVOID; typedef PVOID HANDLE;</code>	Con trỏ kiểu void
SYSTEMTIME	<code>typedef struct _SYSTEMTIME { WORD wYear; WORD wMonth; WORD wDayOfWeek; WORD wDay; WORD wHour; WORD wMinute; WORD wSecond; WORD wMilliseconds; } SYSTEMTIME, *LPSYSTEMTIME;</code>	Thời gian hệ thống, bao gồm 8 trường là năm, tháng, thứ, ngày, giờ, phút, giây và mili giây. Biến kiểu SYSTEMTIME có thể nhận giá trị giờ quốc tế (UTC) hoặc giờ địa phương, tùy thuộc vào hàm được gọi.
FILETIME	<code>typedef struct _FILETIME { DWORD dwLowDateTime; DWORD dwHighDateTime; } FILETIME, *PFILETIME, *LPFILETIME;</code>	Một số 64-bit biểu diễn giá trị thời gian tính từ ngày 1, tháng 1 năm 1601 (UTC) tới thời điểm hiện tại, với đơn vị là 100 ns.
PROCESS_INFORMATION	<code>typedef struct _PROCESS_INFORMATION { HANDLE hProcess; HANDLE hThread; DWORD dwProcessId; HANDLE dwThreadId; } PROCESS_INFORMATION, *PPROCESS_INFORMATION, *LPPROCESS_INFORMATION;</code>	Bao gồm các thông tin về tiến trình và các luồng của tiến trình hProcess : con trỏ đến tiến trình vừa tạo hThread : con trỏ đến các luồng cơ bản của tiến trình vừa tạo dwProcessId : Giá trị để nhận biết tiến trình vừa tạo dwThreadId : Giá trị để nhận biết

		từng luồng của tiến trình vừa tạo
PROCESSENTRY32	<pre>typedef struct tagPROCESSENTRY32 { DWORD dwSize; DWORD dwUsage; DWORD th32ProcessID; ULONG_PTR th32DefaultHeapID; DWORD th32ModuleID; DWORD cntThreads; DWORD th32ParentProcessID; LONG pcPriClassBase; DWORD dwFlags; CHAR szExeFile[MAX_PATH]; } PROCESSENTRY32;</pre>	<p>Các thông tin về một tiến trình trong danh sách các tiến trình trong không gian địa chỉ hệ thống thu được từ snapshot. (<i>Describes an entry from a list of the processes residing in the system address space when a snapshot was taken</i>).</p>
STARTUPINFO	<pre>typedef struct _STARTUPINFOFOW { DWORD cb; LPSTR lpReserved; LPSTR lpDesktop; LPSTR lpTitle; DWORD dwX; DWORD dwY; DWORD dwXSize; DWORD dwYSize; DWORD dwXCountChars; DWORD dwYCountChars; DWORD dwFillAttribute; DWORD dwFlags; WORD wShowWindow; WORD cbReserved2; PBYTE lpReserved2; HANDLE hStdInput; HANDLE hStdOutput; HANDLE hStdError; } STARTUPINFOFOW;</pre>	<p>Cho biết vị trí xuất hiện, màn hình (nếu có nhiều hơn 1 màn hình), các thông số cơ bản (kích thước, màu sắc, các giá trị điều khiển...) liên quan đến quản lý cửa sổ làm việc của một tiến trình khi tiến trình đó được khởi tạo. (<i>Specifies the window station, desktop, standard handles, and appearance of the main window for a process at creation time</i>).</p>
WIN32_FIND_DATA	<pre>typedef struct _WIN32_FIND_DATA { DWORD dwFileAttributes; FILETIME ftCreationTime; FILETIME ftLastAccessTime; FILETIME ftLastWriteTime; DWORD nFileSizeHigh; DWORD nFileSizeLow; DWORD dwReserved0; DWORD dwReserved1; CHAR cFileName[MAX_PATH]; CHAR cAlternateFileName[14]; DWORD dwFileType; DWORD dwCreatorType; WORD wFinderFlags;</pre>	<p>Bao gồm thông tin về file được tìm thấy bởi các hàm <i>FindFirstFile</i>, <i>FindFirstFileEx</i> hoặc <i>FindNextFile</i> (tên, thời gian tạo, lần truy cập cuối, thuộc tính...)</p>

	<pre>} WIN32_FIND_DATAA, *PWIN32_FIND_DATAA, *LPWIN32_FIND_DATAA;</pre>	
<code>LARGE_INTEGER</code>	<pre>typedef struct _LARGE_INTEGER { LONGLONG QuadPart; } LARGE_INTEGER;</pre>	Giá trị đại diện cho một số nguyên 64- bit

3.2 Tổng quan về cách thực hiện chương trình

Sơ đồ thiết kế tổng quan chương trình Tiny-shell:



3.2.1 Xử lý tham số đầu vào

Chương trình là một vòng lặp *do while()* vô hạn, liên tục nhận và thực hiện lệnh và chỉ kết thúc khi người dùng nhập lệnh *exit*.

Shell nhận lệnh người dùng nhập vào từ bàn phím hoặc đọc từ file ***.bat**, tiến hành chuẩn hoá xâu đầu vào (xóa khoảng trắng thừa, đảm bảo xâu đầu vào phải thoả mãn là chữ thường, các tham số chỉ ngăn cách nhau bởi một dấu cách).

Nếu xâu đầu vào là rỗng, thực hiện nhận lệnh tiếp theo, nếu không thì tiến hành phân tích xâu đầu vào thành các phần tử của mảng các tham số *char **argv* (bao gồm các lệnh, tên file

thực thi, tên biến môi trường và các tham số dòng lệnh,...), mỗi phần tử cách nhau bởi dấu cách; đồng thời đếm số phần tử để gán giá trị cho biến *int argc*;

3.2.2 Xử lý lệnh

Kiểm tra phần tử *argv[0]* (chính là lệnh (command) cần thực thi) có đuôi “.bat” không, nếu không có thì thêm đuôi “.bat” vào sau lệnh đó và kiểm tra xem có file *.bat nào cùng tên với lệnh không. Nếu có, gọi hàm thực thi file *.bat là *void ExecuteBatFile(char* filename)*; để thực hiện, nếu không, gọi hàm *void ExecuteCommand(int argc, char** argv, char* command)*;

Hàm *ExecuteCommand()* đề cập ở trên sẽ tiến hành tìm kiếm lệnh vừa nhập trong mảng các câu lệnh mà shell có thể thực thi. Nếu tìm thấy, hàm này sẽ gọi hàm tương ứng để thực thi lệnh vừa nhập. Cuối cùng, nếu lệnh vừa nhập không thuộc các trường hợp kể trên, shell sẽ thực hiện tạo tiến trình **Background** hoặc **Foreground** (phụ thuộc vào tham số đầu vào có dấu ‘&’ hay không). Nếu thất bại trong việc tạo tiến trình, shell sẽ đưa ra thông báo và chuyển sang câu lệnh tiếp theo.

3.3.3 Xử lý file *.bat

Mở file *.bat bằng hàm *fopen()*, sau đó đọc từng câu lệnh bằng hàm *fgets()* và gọi hàm *void RunCommand(char* inputString)* để thực hiện (Hàm *RunCommand()* là một build-in function, chính là hàm sử dụng để xử lý xâu đầu vào và thực hiện lệnh ở trên).

3.3 Tiến trình và các tác vụ liên quan đến tiến trình

3.3.1 Tạo tiến trình Background

Sử dụng hàm *CreateProcess* để tạo tiến trình.

```

BOOL CreateProcessA(
    LPCSTR                lpApplicationName,
    LPSTR                 lpCommandLine,
    LPSECURITY_ATTRIBUTES lpProcessAttributes,
    LPSECURITY_ATTRIBUTES lpThreadAttributes,
    BOOL                  bInheritHandles,
    DWORD                 dwCreationFlags,
    LPVOID                lpEnvironment,
    LPCSTR                lpCurrentDirectory,
    LPSTARTUPINFOA        lpStartupInfo,
    LPPROCESS_INFORMATION lpProcessInformation
);

```

Khi sử dụng hàm này hệ điều hành Windows mặc định sẽ chạy tiến trình theo kiểu

Background. Nếu không tạo được tiến trình sẽ trả về FALSE, trong trường hợp này sẽ xuất ra thông báo câu lệnh vừa nhập không phải câu lệnh của chương trình.

3.3.2 Tạo tiến trình Foreground

Nếu cuối câu lệnh nhập vào có kí tự ‘&’ thì chương trình sẽ thực hiện theo kiểu

Foreground. Sử dụng hàm *CreateProcess()* để tạo tiến trình chạy theo kiểu **Background**.

Ngay sau đó, gọi hàm chờ đợi vô thời hạn *WaitForSingleObject()* đối với tiến trình vừa tạo:

```

DWORD WaitForSingleObject(
    HANDLE hHandle,
    DWORD dwMilliseconds
);

```

Hàm này nhận hai tham số, tham số thứ nhất là con trỏ tới tiến trình **Foreground** cần tạo, tham số thứ hai là thời gian chờ đợi, ở đây ta để giá trị INFINITE.

Để tiến trình nhận tín hiệu kết thúc từ bàn phím (tổ hợp phím **Ctrl + C**), ta sử dụng hàm *SetConsoleCtrlHandler()*:

```
BOOL WINAPI SetConsoleCtrlHandler(
    _In_opt_ PHANDLER_ROUTINE HandlerRoutine,
    _In_      BOOL Add
);
```

và hàm nội tại *CtrlHandler()*:

```
BOOL WINAPI CtrlHandler(
    _In_ DWORD dwCtrlType
);
```

Hàm *SetConsoleCtrlHandler()* cho chương trình biết khi có tín hiệu đầu vào là **Ctrl + C** thì shell sẽ thực hiện hàm *CtrlHandler()*. Hàm này có tham số đầu vào là tín hiệu ngắt **Ctrl + C** và thực hiện huỷ tiến trình *Foreground* đang chạy.

3.3.3 List

Sử dụng hàm *CreateToolhelp32Snapshot()* để tạo một danh sách (snapshot) của các tiến trình đang chạy.

```
HANDLE CreateToolhelp32Snapshot(
    DWORD dwFlags,
    DWORD th32ProcessID
);
```

Dùng hàm *Process32First()* để truy cập vào tiến trình đầu tiên.

```
BOOL Process32First(
    HANDLE hSnapshot,
    LPPROCESSENTRY32 lppe
);
```

Hàm này nhận đầu vào là một con trỏ tới snapshot vừa tạo, và một con trỏ khác kiểu *PROCESSENTRY32* để ghi nhận thông tin về tiến trình đầu tiên.

Để chuyển sang tiến trình tiếp theo gọi hàm *Process32Next()*.

```
BOOL Process32Next(
    HANDLE hSnapshot,
    LPPROCESSENTRY32 lppe
);
```

Lệnh *list* sẽ đưa ra danh sách các tiến trình không được hệ điều hành bảo vệ.

Lệnh *list all* sẽ đưa ra danh sách toàn bộ các tiến trình, bao gồm các tiến trình được hệ điều hành bảo vệ.

Để phân biệt tiến trình được hệ điều hành bảo vệ và tiến trình không được hệ điều hành bảo vệ, sử dụng hàm *OpenProcess()* với quyền truy cập là *PROCESS_ALL_ACCESS*:


```
HANDLE OpenProcess (
    DWORD dwDesiredAccess,
    BOOL bInheritHandle,
    DWORD dwProcessId
);
```

Nếu không được phép truy cập thì đây là tiến trình được hệ điều hành bảo vệ, ngược lại tiến trình này không được hệ điều hành bảo vệ.

3.3.4 Search

Sử dụng hàm *CreateToolhelp32Snapshot()* để tạo một danh sách của các tiến trình đang chạy.

Dùng hàm *Process32First()* để truy cập vào tiến trình đầu tiên, thông tin trả về sẽ được trả về vào một biến có kiểu *PROCESSENTRY32*.

Để chuyển sang tiến trình tiếp theo gọi hàm *Process32Next()*.

Lệnh *seach <process_name>* trả về thông tin của các tiến trình có tên là *process_name* nếu tồn tại, ngược lại thông báo tiến trình không tồn tại.

Lệnh *search pid <process_id>* trả về thông tin của của tiến trình có mã tiến trình là *process_id* nếu tồn tại, ngược lại thông báo tiến trình không tồn tại.

Chương trình được xây dựng thêm hàm nội tại (build-in function) là:

```
PROCESS SearchProcessName (
    char* processName
);
```

và

```
PROCESS SearchProcessID (
    int processID
);
```

lần lượt dùng trả về tiến trình đầu tiên tìm được theo tên và theo mã tiến trình.

3.3.5 Stop

Sử dụng hàm nội tại *SearchProcessName()* và *SearchProcessID()* để tìm kiếm tiến trình.

Có thể sử dụng hàm *TerminateThread()* để dừng lần lượt từng luồng của một tiến trình.

```
BOOL TerminateThread (
    HANDLE hThread,
    DWORD dwExitCode
);
```

Tuy nhiên sử dụng hàm này tiềm ẩn nguy cơ tạo ra bế tắc nếu ta không kiểm soát được thứ tự dừng (và tiếp tục (*resume*) sau đó) của các luồng trong tiến trình. Ta có phương án thay thế là sử dụng hàm *NtSuspendProcess()* bằng cách lấy địa chỉ tương đối của hàm này trong module *ntdll.dll* thông qua hàm *GetProcAddress()*.

```
FARPROC GetProcAddress (
    HMODULE hModule,
    LPCSTR lpProcName
);
```

hModule là một con trỏ tới module *ntdll.dll*, còn *lpProcName* là tên của hàm cần lấy địa chỉ, cụ thể ở đây là hàm *NtSuspendProcess()*.

3.3.6 Resume

Được xây dựng tương tự như lệnh *stop*.

3.3.7 Kill

Sử dụng hàm nội tại *SearchProcessName()* và *SearchProcessID()* để tìm kiếm tiến trình. Sau đó dùng hàm *TerminateProcess()* để kết thúc tiến trình.

```

        BOOL TerminateProcess (
            HANDLE hProcess,
            UINT    uExitCode
        );

```

3.4 Help/Exit/Clear Screen

3.4.1 Help

Kiểm tra tham số đầu vào, nếu thỏa mãn *argc = 1* thì in ra thông tin về các lệnh mà shell hỗ trợ, nếu không thỏa mãn thì thực hiện lệnh tiếp theo.

3.4.2 Exit

Tương tự lệnh *help*, nếu *argc* thỏa mãn thì gọi hàm *exit()*; để kết thúc chương trình.

3.4.3 Clear Screen

Tương tự lệnh *help*, nếu *argc* thỏa mãn thì gọi hàm *system("cls");* để xóa màn hình.

3.5 Directory

Sử dụng hàm *GetCurrentDirectory()* để lấy đường dẫn tuyệt đối đến thư mục hiện tại.

```

        DWORD GetCurrentDirectory (
            DWORD    nBufferLength,
            LPTSTR lpBuffer
        );

```

với *nBufferLength* là độ dài của đường dẫn (đường dẫn là một chuỗi ký tự), bao gồm cả ký tự kết thúc chuỗi ở cuối. *lpBuffer* là một con trỏ tới vùng nhớ sẽ dùng để lưu trữ chuỗi ký tự đường dẫn cần lấy.

Từ đường dẫn tìm được, sử dụng hàm *FindFirstFile()* để tìm file (thư mục con) đầu tiên trong thư mục hiện tại.

```

        HANDLE FindFirstFile (
            LPCSTR          lpFileName,
            LPWIN32_FIND_DATA lpFindFileData
        );

```

Để chuyển sang thư mục tiếp theo sử dụng hàm *FileNextFile()*:

```

        BOOL FindNextFile (
            HANDLE          hFindFile,
            LPWIN32_FIND_DATA lpFindFileData
        );

```

Trong hai hàm trên, *lpFindFileData* là một con trỏ kiểu *WIN32_FIND_DATA*, có các trường cung cấp thông tin về file (thư mục) tìm thấy, *lpFileName* là đường dẫn tới thư mục hiện tại (thư mục cha).

Đặc biệt, đối với các trường thông tin về thời gian tạo, lần truy cập cuối đến file, hàm này trả về thời gian kiểu FILETIME. Để có thể sử dụng được cần phải chuyển đổi về kiểu SYSTEMTIME bằng hàm *FileTimeToSystemTime()*.

```

        BOOL FileTimeToSystemTime (
            const FILETIME *lpFileTime,
            LPSYSTEMTIME  lpSystemTime
        );

```

3.6 Date/Time

3.6.1 Time

Sử dụng hàm *GetLocalTime()*:

```

        void GetLocalTime (
            LPSYSTEMTIME lpSystemTime
        );

```

lpSystemTime là một con trỏ kiểu SYSTEMTIME, con trỏ này sẽ được gán cho các trường của nó giá trị local date and time.

3.6.2 Set time

Sử dụng hàm *SetLocalTime()*:

```

        BOOL SetLocalTime (
            const SYSTEMTIME *lpSystemTime
        );

```

Lệnh *settime* nhận đầu vào là một chuỗi kí tự giờ:phút:giây do người dùng cung cấp, tiến hành tách chuỗi và kiểm tra tính hợp lệ của giá trị nhập vào, sau đó gọi hàm *SetLocalTime()* để thay đổi thời gian. Hàm *SetLocalTime()* chỉ chạy được với quyền **Administrator** nên shell phải có quyền **Administrator** thì hàm mới hoạt động.

3.6.3 Set date

Tương tự lệnh *settime*.

3.7 Path/AddPath

3.7.1 Path

Sử dụng các hàm

- *GetEnvironmentStrings()*:

```

        LPTCH WINAPI GetEnvironmentStrings(void);

```

Hàm này trả về giá trị là con trỏ trỏ tới khối biến môi trường của tiến trình hiện thời, bao gồm cả *system environment variable* và *user environment variable*.

- *lstrlen()*:

```

        int lstrlenA(
            LPCSTR lpString
        );

```

```
);
```

Hàm *lstrlen()* nhận đầu vào là một con trỏ chuỗi và trả về kích thước của chuỗi mà con trỏ đó trỏ tới, không tính phân tử '\0'.

- *FreeEnvironmentStrings()*:

```
BOOL WINAPI FreeEnvironmentStrings(
    _In_ LPCTSTR lpEnvironmentBlock
);
```

Hàm giải phóng bộ nhớ cho 1 con trỏ trỏ tới khối biến môi trường khi không còn cần đến nữa.

3.7.2 Add Path

Sử dụng hàm *SetEnvironmentVariable()*:

```
BOOL SetEnvironmentVariable(
    LPCTSTR lpName,
    LPCTSTR lpValue
);
```

Hàm này nhận đầu vào là 2 con trỏ chuỗi, trỏ đến tên của biến môi trường cần tạo (*lpName*) và giá trị tương ứng (*lpValue*), thực hiện tạo một *user environment variable*.

TÀI LIỆU THAM KHẢO

Kiểu dữ liệu trên Window:

<https://docs.microsoft.com/en-us/windows/desktop/winprog/windows-data-types>

<https://docs.microsoft.com/en-us/windows/desktop/intl/windows-data-types-for-strings>

<https://docs.microsoft.com/en-us/windows/desktop/api/minwinbase/ns-minwinbase-systemtime>

<https://docs.microsoft.com/en-us/windows/desktop/api/tlhelp32/ns-tlhelp32-tagprocessentry32>

<https://docs.microsoft.com/en-us/windows/desktop/api/processthreadsapi/ns-processthreadsapi-startupinfoa>

https://docs.microsoft.com/en-us/windows/desktop/api/minwinbase/ns-minwinbase-win32_find_dataa

https://docs.microsoft.com/en-us/windows/desktop/api/winnt/ns-winnt-large_integer

<https://docs.microsoft.com/en-us/windows/desktop/api/minwinbase/ns-minwinbase-filetime>

[https://msdn.microsoft.com/zh-cn/dynamics/ms684873\(v=vs.90\)](https://msdn.microsoft.com/zh-cn/dynamics/ms684873(v=vs.90))

Các hàm API:

<https://docs.microsoft.com/en-us/windows/desktop/api/timezoneapi/nf-timezoneapi-filetimetosystemtime>

<https://docs.microsoft.com/en-us/windows/desktop/api/timezoneapi/nf-timezoneapi-systemtimetotzspecificlocaltime>

<https://docs.microsoft.com/en-us/windows/desktop/api/sysinfoapi/nf-sysinfoapi-getlocaltime>

<https://docs.microsoft.com/en-us/windows/desktop/api/sysinfoapi/nf-sysinfoapi-setlocaltime>

<https://docs.microsoft.com/en-us/windows/desktop/toolhelp/taking-a-snapshot-and-viewing-processes>

<https://docs.microsoft.com/en-us/windows/desktop/api/synchapi/nf-synchapi-waitforsingleobject>

<https://docs.microsoft.com/en-us/windows/desktop/api/winbase/nf-winbase-getcurrentdirectory>

<https://docs.microsoft.com/en-us/windows/desktop/api/fileapi/nf-fileapi-findfirstfilea>

<https://docs.microsoft.com/en-us/windows/desktop/api/fileapi/nf-fileapi-findnextfilea>

<https://docs.microsoft.com/en-us/windows/console/setconsolectrlhandler>

<https://docs.microsoft.com/en-us/windows/desktop/api/tlhelp32/nf-tlhelp32-createtoolhelp32snapshot>

<https://docs.microsoft.com/en-us/windows/desktop/api/tlhelp32/nf-tlhelp32-process32first>

[https://msdn.microsoft.com/en-us/library/windows/desktop/ms683187\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/desktop/ms683187(v=vs.85).aspx)

[https://msdn.microsoft.com/en-us/library/windows/desktop/ms683151\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/desktop/ms683151(v=vs.85).aspx)

<https://docs.microsoft.com/en-us/windows/desktop/api/winbase/nf-winbase-setenvironmentvariable>

<https://docs.microsoft.com/en-us/windows/desktop/api/winbase/nf-winbase-lstrlena>

<https://docs.microsoft.com/en-us/windows/desktop/api/processthreadsapi/nf-processthreadsapi-terminatethread>

<https://docs.microsoft.com/en-us/windows/desktop/api/libloaderapi/nf-libloaderapi-getProcAddress>

Tài liệu khác:

William Stallings, Operating System: Internals and Design Principles – 7th Edition, Prentice Hall, 2012.

<https://github.com/quanvuhust/TinyShell>

<https://docs.microsoft.com/en-us/windows/desktop/apiindex/api-index-portal>

[https://en.wikipedia.org/wiki/Shell_\(computing\)](https://en.wikipedia.org/wiki/Shell_(computing))

https://www.ibm.com/support/knowledgecenter/en/ssw_aix_71/com.ibm.aix.osdevice/shells.htm

<https://www.computerhope.com/jargon/e/envivari.htm>