Môi trường hoạt động của process

Giảng Viên: Lưu An Phú

Agenda

- Điểm bắt đầu và kết thúc của process
- Tham số truyền vào và biến môi trường
- Cấu trúc bộ nhớ của process
- Cách process sử dụng shared library.
- Terminal login, process group, process session.
- Background and force ground process.

Điểm bắt đầu và kết thúc của process

Điểm bắt đầu và kết thúc

- Hàm main
  - Điểm đầu tiên đi vào chương trình
    - *Nếu chúng ta khai báo thiếu các tham số thì trình biên dịch cũng sẽ tự động thêm vào*
  - Khi run chương trình, hệ điều hành sẽ chạy một số đoạn code ẩn dựa vào các thông số của chương trình, những đoạn code đó nằm ngoài code của chúng ta.

Điểm bắt đầu và kết thúc

- Kết thúc chương trình
  - Là điểm cuối cùng trước khi đi ra khỏi chương trình
  - Có nhiều cách để kết thúc chương trình
    - Chủ động kết thúc
    - Bị động kết thúc
  - Giá trị trả về của chương trình khi kết thúc có thể nhận được từ chương trình cha.

Biến môi trường

Command line argument

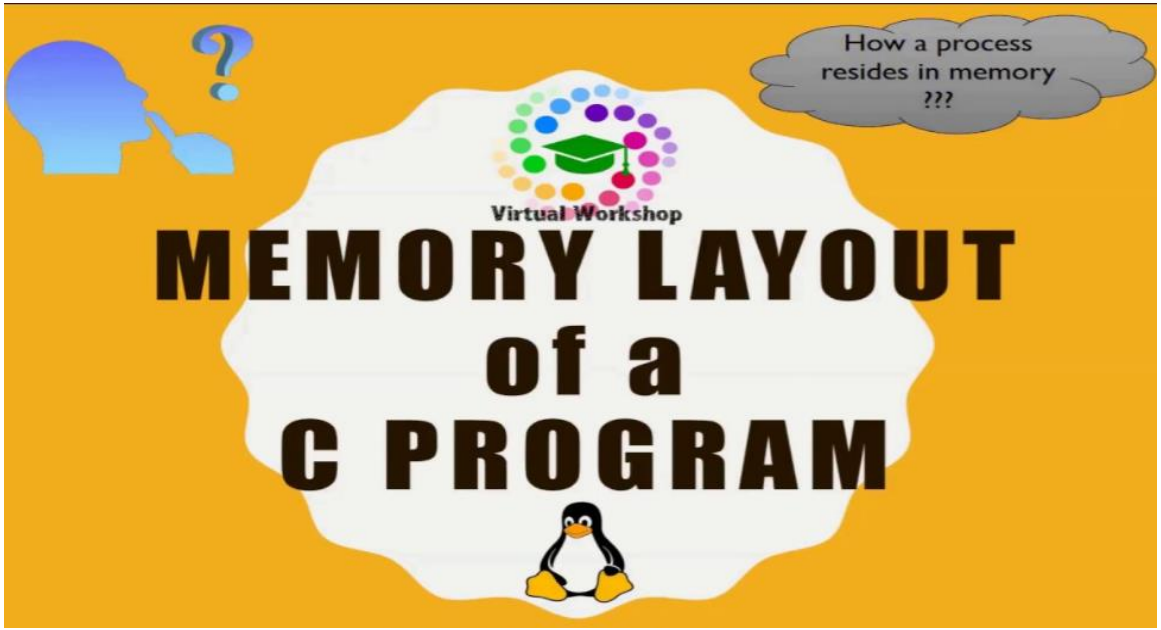
```
ne-test:/$ ls -l

root  root   4096 Th10  4 13:04 bin
root  root   4096 Th10  4 13:14 boot
root  root   4096 Th03  8  2018 cdrom
root  root   4096 Th04 12  2018 data
root  root   4100 Th11 21 11:59 dev
root  root  12288 Th11 21 14:02 etc
phula phula  4096 Th05  3  2018 home
```

- Chúng là 2 biến argc và argv[] của hàm main
- argc là số tham số truyền vào, argv là các chuỗi ký tự truyền vào khi gọi hàm chương trình

Environment list

```
phula@alb-machine-test:/$ printenv
LC_PAPER=vi_VN
LC_ADDRESS=vi_VN
XDG_SESSION_ID=123
LC_MONETARY=vi_VN
TERM=xterm
SHELL=/bin/bash
SSH_CLIENT=192.168.137.1 61433 22
LC_NUMERIC=vi_VN
SSH_TTY=/dev/pts/7
http_proxy=http://phula:0105#Day@10.16.29.21:8080
USER=phula
LC_TELEPHONE=vi_VN
ftp_proxy=http://phula:0105#Day@10.16.29.21:8080
all_proxy=sock://phula:0105#Day@10.16.29.21:8080
ALL_PROXY=http://phula:0105#Day@10.16.29.21:8080
MAIL=/var/mail/phula
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr
QT_QPA_PLATFORMTHEME=appmenu-qt5
LC_IDENTIFICATION=vi_VN
```

- Store in a global variable extern char **environ
- Có thể truy xuất thông qua hàm char *getenv(const char *name)
- Thực hành
  - Hiển thị tên và giá trị của các biến môi trường bằng C



Cấu trúc bộ nhớ của process

Cấu trúc bộ nhớ của process

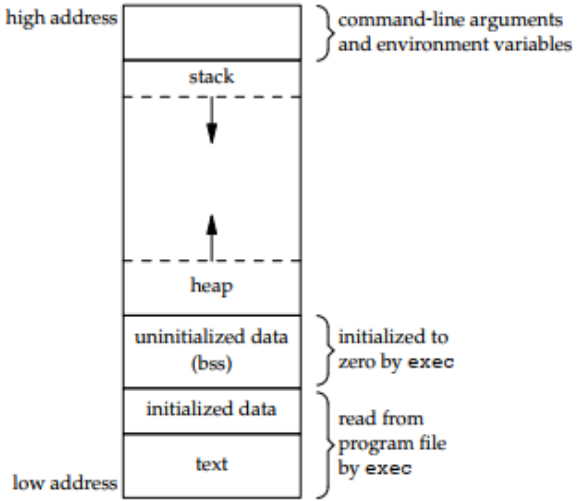


Figure 7.6 Typical memory arrangement

- Vùng text
- Vùng dữ liệu đã được khởi tạo
- Vùng dữ liệu chưa được khởi tạo
- Vùng stack
- Vùng heap
  - void *malloc(size_t size);

Shared library

- Là các file binary được build từ code C
- Chúng chứa các hàm dùng để gọi từ chương trình khác
- Mỗi khi cần truy xuất hàm từ thư viện động, hệ điều hành sẽ load 1 phần hoặc toàn bộ thư viện đó vào ram, sau đó trả về địa chỉ hàm cho process. Process sẽ gọi hàm thông qua địa chỉ

Bài tập

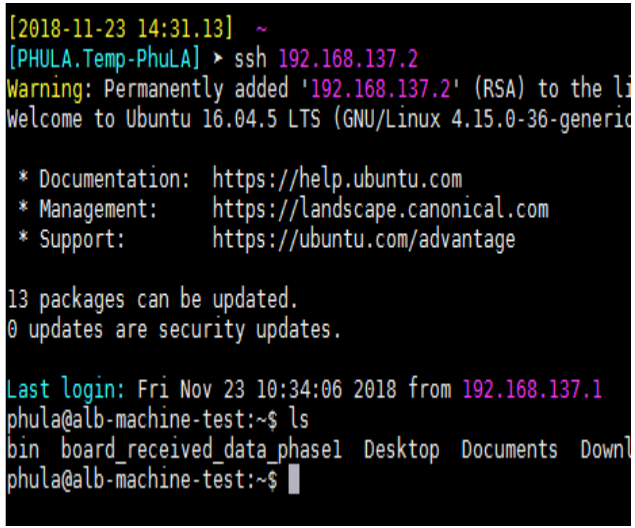
- Viết 1 thư viện rồi build thành shared library, shared lib được copy vào thư mục /usr/lib. Sau đó viết 1 chương trình C sử dụng shared lib đó.
- Viết 1 chương trình C mở 1 file bất kỳ, nếu mở thành công thì retrun 0 cho chương trình cha, nếu lỗi thì return -1 cho chương trình cha. Dùng terminal để chạy và echo vào biến $ để đọc kết quả trả về

Bài tập

- Viết một chương trình xử lý tham số đầu vào giống với câu lệnh ls. Ví dụ nếu call "my_ls -l" sẽ in ra dòng chữ hiện thị full property, nếu truyền -a sẽ in hiển thị file ẩn, nếu truyền -la sẽ in ra cả 2 dòng trên.
- Viết 1 chương trình C truy cập vào environment variable và hiển thị đường dẫn thư mục home, tên user, đường dẫn hiện tại của chương trình.

Terminal

Terminal login



- Mỗi 1 chương trình khi start đều cần được gán các luồng STDIN, STDOUT và STDERR cho nó
- Một terminal sẽ bao gồm 3 luồng nhập xuất tiêu chuẩn kể trên.
- Trong chế độ console, mỗi khi log in thành công thì 1 terminal device sẽ được tạo ra, các chương trình được run bởi user đó sẽ gắn với terminal login của user.

Background and force ground process

- Forceground process
  - Process đang kết nối đến STDIN của terminal
- Background process
  - Process ngắt kết nối đến STDIN của terminal
  - Ký tự ”&” khi run command line

Process group

- Các process làm chung 1 loại công việc có thể được gom vào cùng 1 process group
- Dùng để chuyển đổi force và background mode cùng với nhau
- Process group đại diện bởi 1 số unsigned int
  - pid_t getpgid(pid_t pid);
  - int setpgid(pid_t pid, pid_t pgid);
- Khi 1 process được tạo ra, mặc định nó sẽ cùng group với process cha
- Signal có thể được gửi cho tất cả các process trong cùng 1 group

Sesions

- Là tập hợp của nhiều process group có chung 1 môi trường hoạt động
- Có cùng liên kết đến 1 terminal login
- Khi 1 user log out, các process có chung session với user đó sẽ bị exit

Bài tập

- Viết 3 chương trình A, B, C sao cho khi được chạy chúng sẽ sleep 10s rồi print process group id của mình. Run A, B, C bằng câu lệnh sau; ./A | ./B | ./C & và xem kết quả
- Viết 3 chương trình A, B, C print session id, run chúng trên cùng 1 cửa sổ terminal và trên 2 cửa sổ terminal và xem kết quả
- Viết 3 chương trình A, B, C đọc 1 ký tự từ bàn phím, mở chúng bằng 3 terminal khác nhau. Viết 1 chương trình C thứ 4 để nhập ký tự cho 3 chương trình A B C.

Thank you