


TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP THỰC PHẨM TP.HCM
KHOA CÔNG NGHỆ THÔNG TIN

TÀI LIỆU
THỰC HÀNH HỆ ĐIỀU HÀNH

Biên soạn: Bộ môn Mạng máy tính & Truyền thông

MỤC LỤC

BÀI 1. CÁC LỆNH CƠ BẢN LINUX: CÀI ĐẶT, HƯỚNG DẪN SỬ DỤNG	3
BÀI 2. BIÊN DỊCH VỚI GCC	19
BÀI 3. TIẾN TRÌNH	33
BÀI 4. TIỂU TRÌNH	41
BÀI 5. GIẢI THUẬT ĐỊNH THỜI.....	51
BÀI 6. ĐỒNG BỘ HÓA TIẾN TRÌNH	56
BÀI 7. TẮC NGHẼN	63
BÀI 8. QUẢN LÝ BỘ NHỚ.....	66
BÀI 9. QUẢN LÝ BỘ NHỚ ẢO	76
BÀI 10. TẬP TIN	89

Trường: ĐH CNTP TP.HCM Khoa: Công nghệ Thông tin Bộ môn: Mạng máy tính – Truyền thông MH: TH Hệ điều hành MSMH:	<p style="text-align: center;">BÀI 1. CÁC LỆNH CƠ BẢN LINUX: CÀI ĐẶT, HƯỚNG DẪN SỬ DỤNG</p>	
--	--	---

A. MỤC TIÊU:

- Trình bày được trình tự các bước cài đặt hệ điều hành Linux.
- Sử dụng được hệ điều hành Linux ở giao diện đồ họa.
- Sử dụng được các lệnh cơ bản trên Linux
- Mô phỏng và kiểm chứng được các nội dung lý thuyết đã học.

B. DỤNG CỤ - THIẾT BỊ THỰC HÀNH CHO MỘT SV:

STT	Chủng loại – Quy cách vật tư	Số lượng	Đơn vị	Ghi chú
1	Computer	1	1	

C. VẬT LIỆU : Dùng phần mềm CentOS , VMWare.

D. NỘI DUNG THỰC HÀNH

I. Thực hiện cài đặt hệ điều hành CentOS Linux theo một số yêu cầu như sau:

- Cài đặt trực tiếp từ CDROM.
- Chọn “Installation Type” dạng Server.
- Dung lượng đĩa cần thiết để cài đặt khoảng 8GB.
- Cấu hình mạng :

Hostname : ServerTênSV

IP address: 192.168.1.số máy/24

II. Thực hiện một số lệnh cơ bản trên Linux

1. HƯỚNG DẪN THỰC HÀNH

Cài đặt hệ điều hành Linux

Để cài đặt CentOS 7 trước tiên cần download file CentOS-7-x86_64-DVD-2003.iso

Bước 1: Sử dụng VMWARE để cài đặt Centos 7

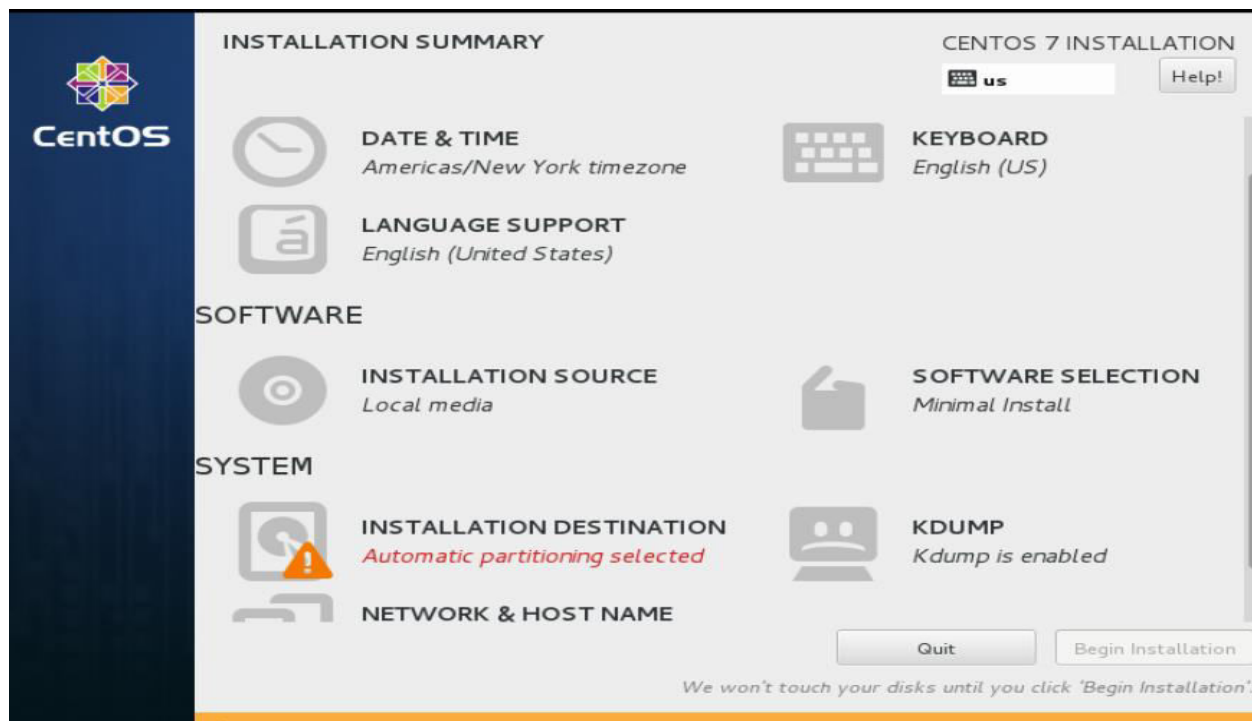
Bước 2: Chọn **Install CentOS 7**



Bước 3: Language: ta chọn English sau đó nhấn Continue để tiếp tục.



Bước 4: Giao diện cấu hình Date & Time , Language , Network & Hostname , Installation Destination , Software Selection. Chọn cài đặt ngày tháng và múi giờ, Software Selection (chọn GNOME desktop), chọn Network and Hostname



Bước 5: Chọn đặt ngày tháng và múi giờ , click **Done**



Bước 6: Chọn Software installation , chọn GNOME Desktop, click **Done**

SOFTWARE SELECTION

CENTOS 7 INSTALLATION

Done

us

Help!

Base Environment

☐ Minimal Install
Basic functionality.

☐ Compute Node
Installation for performing computation and processing.

☐ Infrastructure Server
Server for operating network infrastructure services.

☐ File and Print Server
File, print, and storage server for enterprises.

☐ Basic Web Server
Server for serving static and dynamic internet content.

☐ Virtualization Host
Minimal virtualization host.

☐ Server with GUI
Server for operating network infrastructure services, with a GUI.

☒ GNOME Desktop
GNOME is a highly intuitive and user friendly desktop environment.

☐ KDE Plasma Workspaces
The KDE Plasma Workspaces, a highly-configurable graphical user interface which includes a panel, desktop, system icons and desktop widgets, and many powerful KDE applications.

Add-Ons for Selected Environment

☐ Backup Client
Client tools for connecting to a backup server and doing backups.

☐ GNOME Applications
A set of commonly used GNOME Applications.

☐ Internet Applications
Email, chat, and video conferencing software.

☐ Legacy X Window System Compatibility
Compatibility programs for migration from or working with legacy X Window System environments.

☐ Office Suite and Productivity
A full-purpose office suite, and other productivity tools.

☐ Smart Card Support
Support for using smart card authentication.

☐ Compatibility Libraries
Compatibility libraries for applications built on previous versions of CentOS Linux.

☐ Development Tools
A basic development environment.

☐ Security Tools
Security tools for integrity and trust verification.

☐ System Administration Tools
Utilities useful in system administration.

Bước 7: Chọn Installation Destination

INSTALLATION DESTINATION

CENTOS 7 INSTALLATION

Done

us

Help!

Device Selection

Select the device(s) you'd like to install to. They will be left untouched until you click on the main menu's "Begin Installation" button.

Local Standard Disks

20 GiB

VMware, VMware Virtual S
 sda / 20 GiB free

Specialized & Network Disks

Add a disk...

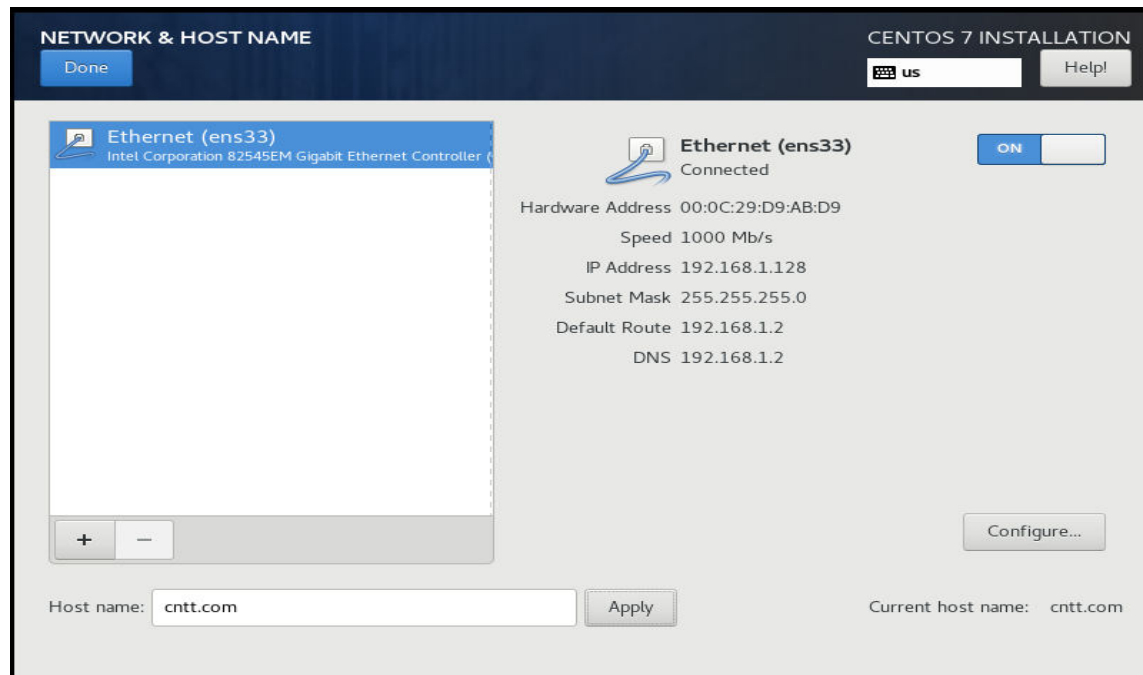
Other Storage Options

Partitioning

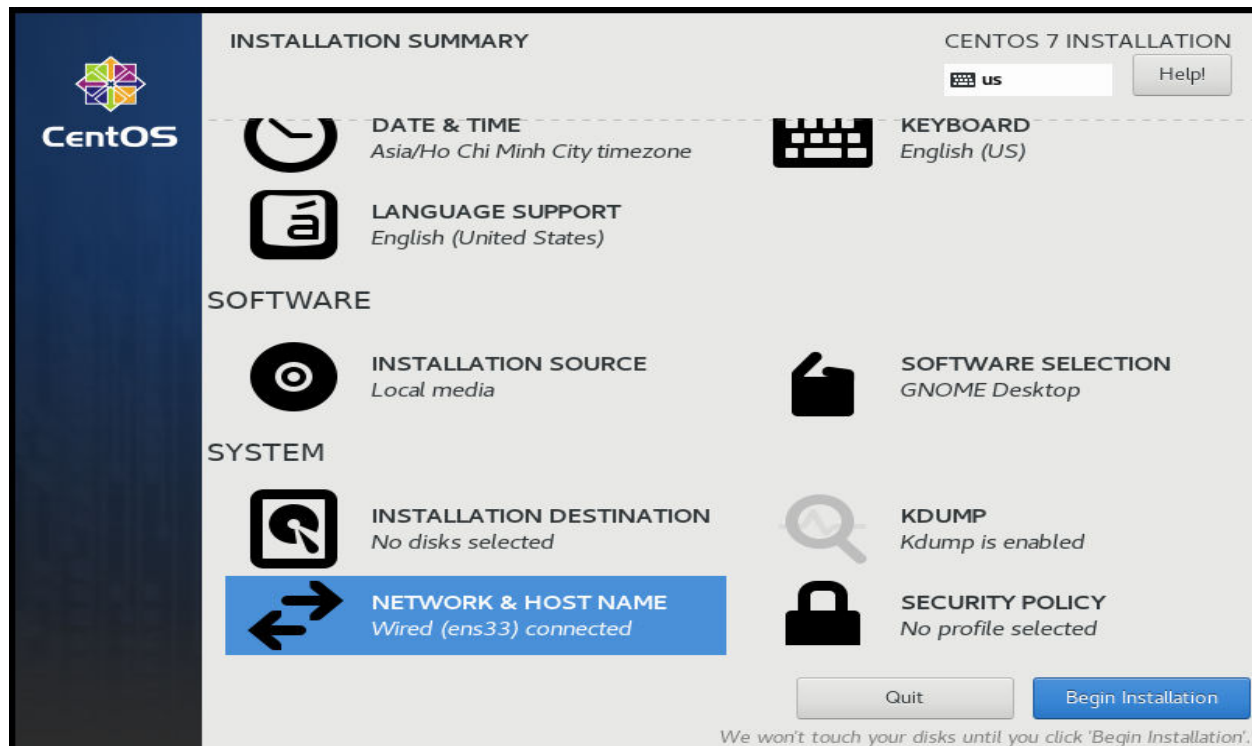
☒ Automatically configure partitioning
 ☐ I will configure partitioning

0 disks selected; 0 B capacity; 0 B free [Refresh...](#)

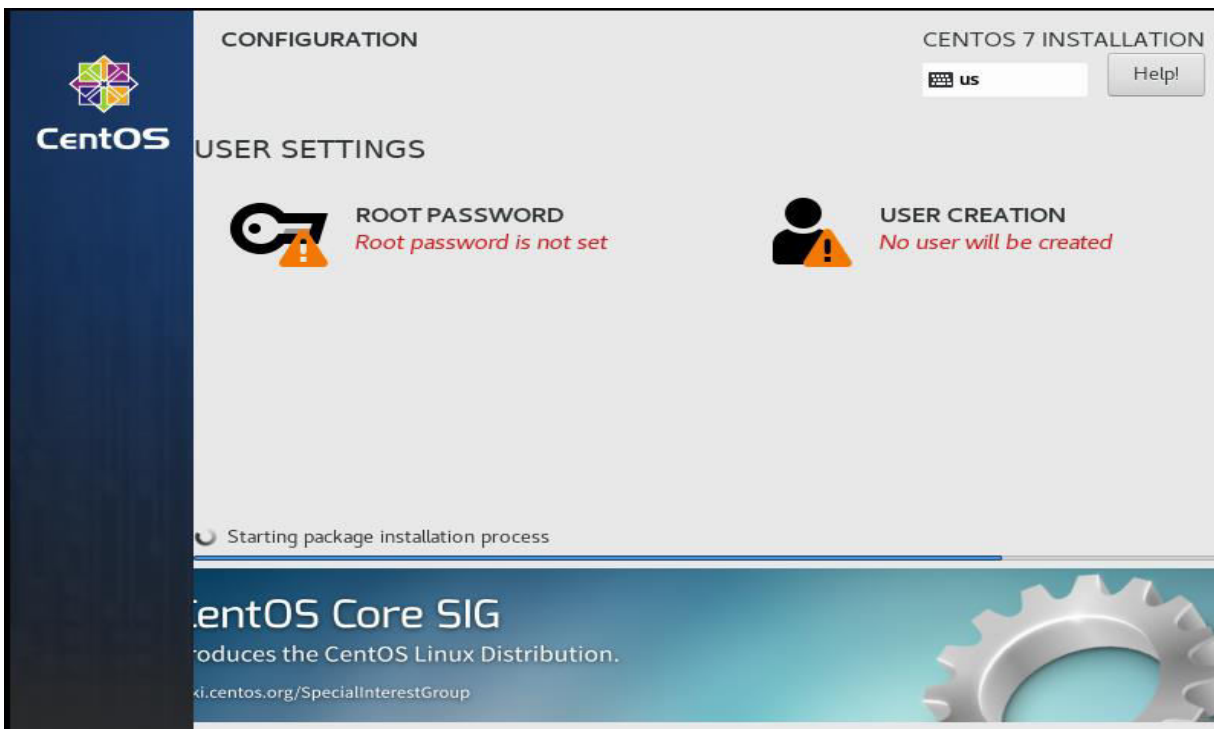
Bước 8: Chọn Network and Hostname , chỉnh sửa Hostname



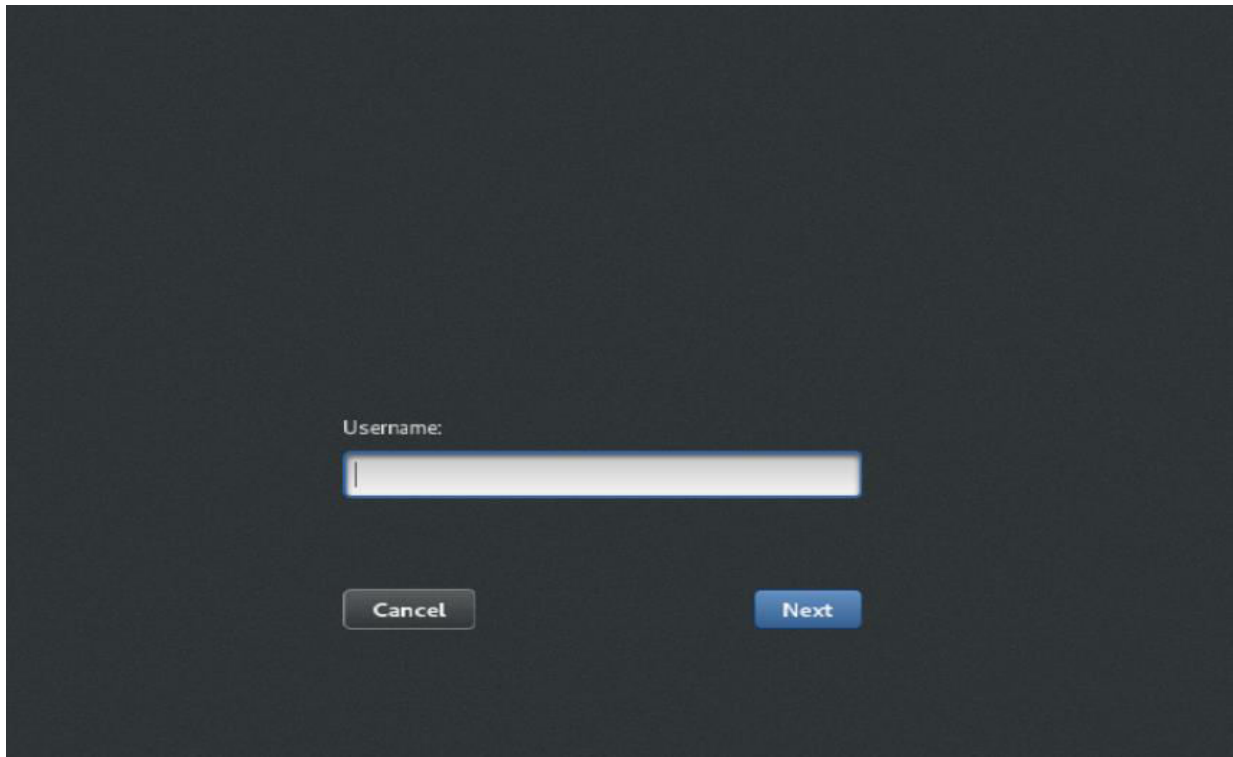
Bước 9: Trở lại trang tóm tắt cài đặt và click “Begin Installation”



Bước 10: Tiến trình cài đặt bắt đầu , đặt **password** cho **root** và tạo **User**



Bước 11: Chờ quá trình cài đặt kết thúc và đăng nhập.



2. Thực hiện một số lệnh cơ bản trên Linux

2.1. Một số chú ý:

- Lệnh và thông số được phân biệt chữ in và chữ thường.
- Mỗi lệnh được gõ trên 1 dòng, nếu dùng nhiều lệnh liên tục thì các lệnh phải được cách nhau bằng dấu ';'.
- Xem help của lệnh (#[man](#) command)

Vd:

```
#man rpm
```

```
:/remove    tìm kiếm chuỗi remove
```

```
n          tìm tiếp tục (tìm xuôi)
```

```
N          tìm ngược
```

```
q          thoát khỏi man
```

1. Giới thiệu dấu nhắc:

[root @ localhost ~] #

root: user

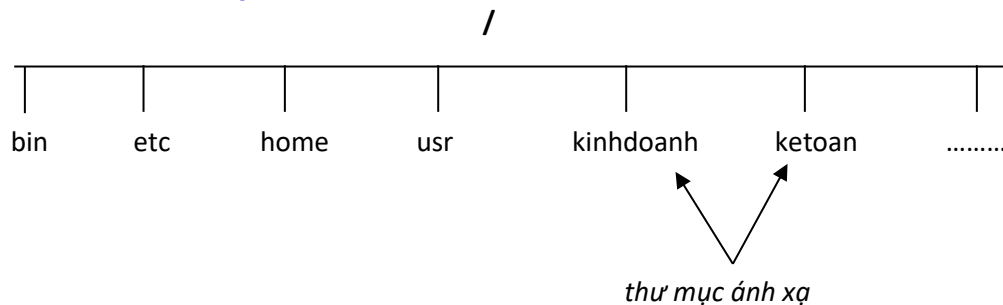
localhost: computer name

~: thư mục cá nhân

#: user toàn quyền; \$: user có quyền giới hạn

2. Nhóm lệnh quản lý thư mục:

- Cây thư mục mặc định do hệ thống tạo ra:



cd: chuyển thư mục làm việc.

mkdir: tạo thư mục

ll: liệt kê (ls)

rmdir: xóa thư mục rỗng

rm: xóa thư mục & cây thư mục

mv: đổi tên (nếu cùng đường dẫn) hoặc di chuyển (nếu khác đường dẫn)

tree: xem cây thư mục

- Tạo nhiều thư mục liên tục :

```
#mkdir -p ../linux/ {fedora,centos,suse}
```

-p : tạo nhiều thư mục

- Đổi tên thư mục os thành hdh

```
#mv /data/os /data/hdh
```

- Di chuyển thư mục software vào trong data

```
#mv /data/dulieu/software /data
```

- Xóa thư mục aix

```
# rmdir /data/hdh/unix/aix
```

- Xóa cây thư mục windows

```
#rm -rf /data/hdh/windows
```

- Xem dung lượng thư mục data

```
#du /data
```

```
#du -h /data : hiển thị đơn vị đo
```

```
#du -h /data | more : ngắt trang
```

- Liệt kê thư mục /bin

```
#ll /bin ⇔ #ls -l /bin
```

```
#ls /bin
```

```
#ls -ld /bin : xem số thư mục con
```

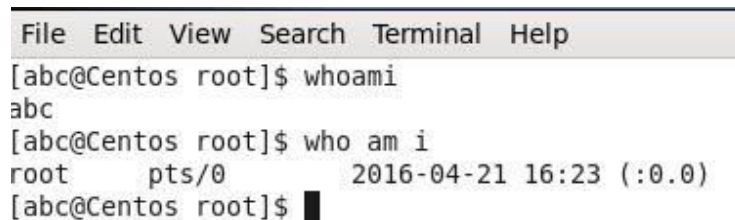
```
# ls -la /bin : xem thư mục ẩn
```

(Thư mục, tập tin có dấu chấm phía trước là tập tin, thư mục ẩn.)

2.2 Bài tập mẫu

Yêu cầu: Không sử dụng giao diện đồ họa, sử dụng command line để thực hiện các lệnh cơ bản

1. Login vào hệ thống. Sử dụng lệnh “whoami” hoặc “who am i” để cho biết user hiện tại là gì?



```
File Edit View Search Terminal Help
[abc@Centos root]$ whoami
abc
[abc@Centos root]$ who am i
root pts/0 2016-04-21 16:23 (:0.0)
[abc@Centos root]$
```

2. Dùng lệnh su để thay đổi người sử dụng.

Ví dụ : chuyển người sử dụng là abc qua root

```
File Edit View Search Termina
[abc@Centos root]$ su root
Password:
[root@Centos ~]# █
```

3. Cho biết tên của hệ điều hành đang sử dụng.

```
[root@Centos ~]# uname
Linux
[root@Centos ~]# █
```

4. Cho biết version của kernel của HĐH hiện tại

```
[root@Centos ~]# uname -r
2.6.32-573.el6.x86_64
[root@Centos ~]# █
```

5. Cho biết bao nhiêu user đang login vào hệ thống

```
[root@Centos ~]# w
10:54:39 up 59 min, 2 users, load average: 0.00, 0.01, 0.03
USER      TTY      FROM          LOGIN@      IDLE   JCPU   PCPU WHAT
root      tty1     :0             09:59      58:59  15.29s 15.29s /usr/bin/Xorg :
root      pts/0    :0.0           10:48       0.00s  0.17s  0.17s w
[root@Centos ~]# █
```

6. Cho biết ngày tháng năm hiện tại của hệ thống

```
[root@Centos ~]# date
Fri Apr 22 10:56:41 ICT 2016
[root@Centos ~]# █
```

7. Hiển thị lịch của tháng hiện tại

```
[root@Centos ~]# cal
      April 2016
Su Mo Tu We Th Fr Sa
                1  2
 3  4  5  6  7  8  9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30

[root@Centos ~]# █
```

8. Hiển thị lịch của năm 2016, lịch của tháng 4 năm 2016

```
[root@Centos ~]# cal 4 2016
      April 2016
Su Mo Tu We Th Fr Sa
                1  2
 3  4  5  6  7  8  9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30

[root@Centos ~]# cal 2016
                        2016

      January          February          March
Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa
                1  2          1  2  3  4  5  6          1  2  3  4  5
 3  4  5  6  7  8  9       7  8  9 10 11 12 13       6  7  8  9 10 11 12
10 11 12 13 14 15 16     14 15 16 17 18 19 20     13 14 15 16 17 18 19
17 18 19 20 21 22 23     21 22 23 24 25 26 27     20 21 22 23 24 25 26
24 25 26 27 28 29 30     28 29                   27 28 29 30 31
31

      April           May           June
Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa
                1  2          1  2  3  4  5  6  7          1  2  3  4
 3  4  5  6  7  8  9       8  9 10 11 12 13 14       5  6  7  8  9 10 11
10 11 12 13 14 15 16     15 16 17 18 19 20 21     12 13 14 15 16 17 18
17 18 19 20 21 22 23     22 23 24 25 26 27 28     19 20 21 22 23 24 25
24 25 26 27 28 29 30     29 30 31                 26 27 28 29 30
```

9. Cho biết ngày 1 tháng 1 năm 2017 là thứ mấy?

```
[root@Centos ~]# cal 1 1 2017
      January 2017
Su Mo Tu We Th Fr Sa
 1  2  3  4  5  6  7
 8  9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30 31
```

10. Lệnh pwd cho biết thư mục hiện hành.

```
File Edit View Search Terminal Help
[root@Centos ~]# pwd
/root
[root@Centos ~]#
```

11. Liệt kê danh sách file, folder trong thư mục hiện hành

Hoặc có thể dùng các lệnh sau:

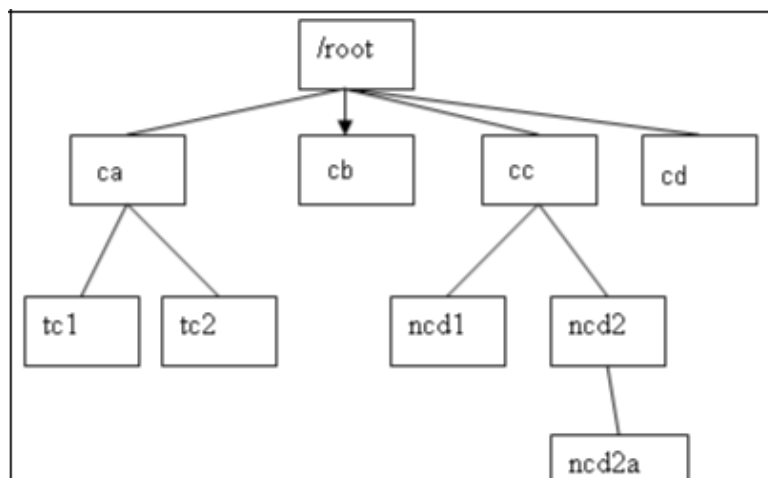
```
File Edit View Search Terminal Help
[root@Centos ~]# ls
Desktop Documents Downloads Music Pictures Public Templates Videos
[root@Centos ~]#
```

ls -x hiện thị nội dung

ls -l hiện thị chi tiết các thông tin của tập tin

ls -a hiện thị tất cả các tập tin kể cả tập tin ẩn

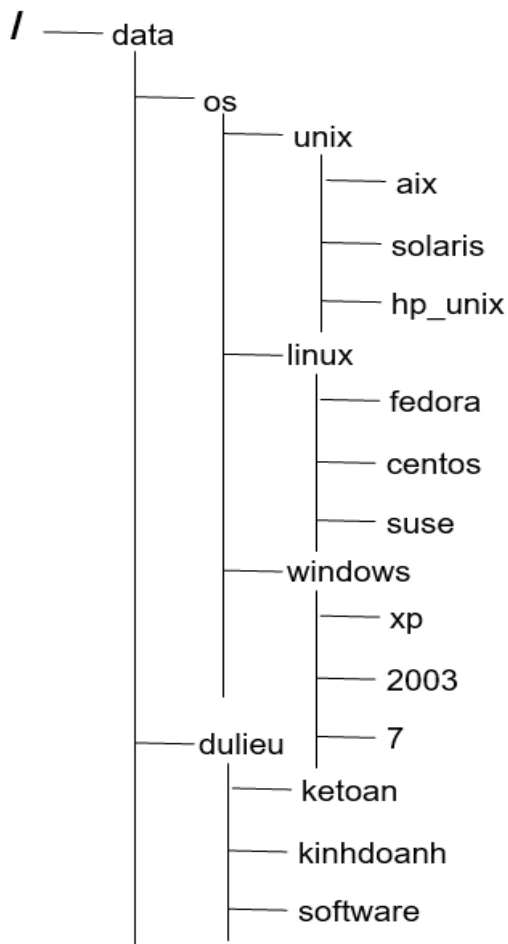
12. Dùng lệnh “mkdir” để tạo cấu trúc cây thư mục như hình, sau đó dùng lệnh “ls” để kiểm tra lại.



2.3. Bài tập luyện tập:

Tạo cây thư mục

```
File Edit View Search Terminal Help
[root@Centos ~]# mkdir /root/ca
[root@Centos ~]# mkdir /root/cb
[root@Centos ~]# mkdir /root/cc
[root@Centos ~]# mkdir /root/ca/tc1
[root@Centos ~]# mkdir /root/ca/tc2
[root@Centos ~]# mkdir /root/cc/ncd1
[root@Centos ~]# mkdir /root/cc/ncd2
[root@Centos ~]# mkdir /root/cc/ncd2/ncd2a
[root@Centos ~]# ls
ca  cc      Documents  Music      Public     Videos
cb  Desktop Downloads  Pictures   Templates
[root@Centos ~]# ls /root/ca
tc1  tc2
[root@Centos ~]# ls /root/cb
[root@Centos ~]# ls /root/cc
ncd1  ncd2
[root@Centos ~]# ls /root/cc/ncd2
ncd2a
[root@Centos ~]# █
```





E. TIÊU CHÍ ĐÁNH GIÁ BÀI THỰC HÀNH

Tiêu chí đánh giá	Mức độ 1	Mức độ 2	Mức độ 3
1. Thời gian	Đúng giờ	Trễ 5 phút	Trễ hơn 5 phút
1 điểm	1	0,6	0,3
2. Nội dung	<i>Đạt yêu cầu</i>	Vài chi tiết chưa đạt yêu cầu	<i>Không đạt yêu cầu</i>
7 điểm	7	5	3,5
3. Báo cáo	Đầy đủ yêu cầu	Vài nội dung còn thiếu	<i>Không đạt yêu cầu</i>
2 điểm	2	1	0,5

F. BÀI TẬP VỀ NHÀ

- Cài đặt hệ điều hành Fedora và Ubuntu
- Thực hiện lại Bài tập mẫu trên Fedora và Ubuntu

Trường: ĐH CNTP TP.HCM Khoa: Công nghệ Thông tin Bộ môn: Mạng máy tính – Truyền thông MH: TH Hệ điều hành MSMH:	<h2 style="text-align: center;">BÀI 2. BIÊN DỊCH VỚI GCC</h2>	
---	---	---

A. MỤC TIÊU

Sau khi hoàn thành bài thực hành này, sinh viên sẽ đạt được các kỹ năng sau:

Biên dịch một chương trình C từ mã nguồn bằng cách sử dụng GNU (tên đầy đủ là GNU Compiler Collection, viết tắt là GCC) - trình biên dịch dành cho Linux và Minimalist Gnu (MinGW) trên Windows.

B. DỤNG CỤ - THIẾT BỊ THỰC HÀNH CHO MỘT SINH VIÊN:

STT	Chủng loại – quy cách vật tư	Số lượng	Đơn vị	Ghi chú
1	Computer	1	Bộ	

C. NỘI DUNG THỰC HÀNH

I. Lý thuyết

1. Chương trình trên Linux

– Để có thể viết chương trình trên Linux, chúng ta cần phải nắm rõ 1 số vị trí tài nguyên để xây dựng chương trình như trình biên dịch, tập tin thư viện, các tập tin tiêu đề (header), các tập tin chương trình sau khi biên dịch, ...

– Trình biên dịch **gcc** thường được đặt trong thư mục **/usr/bin** hoặc **/usr/local/bin** (kiểm tra bằng lệnh **which gcc**). Tuy nhiên, khi biên dịch, **gcc** cần đến rất nhiều tập tin hỗ trợ nằm trong những thư mục khác nhau như những tập tin tiêu đề (header) của C thường nằm trong thư mục **/usr/include** hay **/usr/local/include**. Các tập tin thư viện liên kết thường được **gcc** tìm trong thư mục

`/lib` hoặc `/usr/local/lib`. Các thư viện chuẩn của **gcc** thường đặt trong thư mục `/usr/lib/gcc-lib`.

Chương trình sau khi biên dịch ra tập tin thực thi (dạng nhị phân) có thể đặt bất cứ vị trí nào trong hệ thống.

2. Các tập tin tiêu đề (header)

Các tập tin tiêu đề trong C thường định nghĩa hàm và khai báo cần thiết cho quá trình biên dịch. Hầu hết các chương trình trên Linux khi biên dịch sử dụng các tập tin

tiêu đề trong thư mục `/usr/include` hoặc các thư mục con bên trong thư mục này, ví dụ:

`/usr/include/sys`.

Một số khác được trình biên dịch dò tìm mặc định như `/usr/include/X11` đối với các khai báo hàm lập trình đồ họa X-Window, hoặc `/usr/include/g++-2` đối với trình biên dịch GNU g++.

Tuy nhiên, nếu chúng ta có các tập tin tiêu đề của riêng mình trong một thư mục khác thư mục mặc định của hệ thống thì chúng ta có thể chỉ rõ tường minh đường dẫn đến thư mục khi biên dịch bằng tùy chọn `-I`, ví dụ:

```
$ gcc -I/usr/mypro/include test.c -o test
```

Khi chúng ta sử dụng một hàm nào đó của thư viện hệ thống trong chương trình C, ngoài việc phải biết khai báo nguyên mẫu của hàm, chúng ta cần phải biết hàm này được định nghĩa trong tập tin tiêu đề nào. Trình man sẽ cung cấp cho chúng ta các thông tin này rất chi tiết. Ví dụ, khi dùng man để tham khảo thông tin về hàm `kill()`, chúng ta sẽ thấy rằng cần phải khai báo 2 tập tin tiêu đề là `types.h` và `signal.h`.

3. Các tập tin thư viện

– Các tập tin tiêu đề của C chỉ cần thiết để trình biên dịch bắt lỗi cú pháp, kiểm tra kiểu dữ liệu của chương trình và tạo ra các tập tin đối tượng. Muốn tạo ra chương trình thực thi, chúng ta cần phải có các tập tin thư viện. Trong Linux, các tập tin thư viện tĩnh của C có phần mở rộng là .a, .so, .sa và bắt đầu bằng tiếp đầu ngữ lib. Ví dụ libutil.a hay libc.so là tên các thư viện liên kết trong Linux.

– Linux có hai loại liên kết là liên kết tĩnh (static) và liên kết động (dynamic). Thư viện liên kết động trên Linux thường có phần mở rộng là .so, chúng ta có thể dùng lệnh `ls /usr/lib` hoặc `ls /lib` để xem các thư viện hệ thống đang sử dụng.

Khi biên dịch, thông thường trình liên kết (ld) sẽ tìm thư viện trong 2 thư viện chuẩn `/usr/lib` và `/lib`. Để chỉ định tường minh một thư viện nào đó, chúng ta làm như sau:

```
$ gcc test.c -o test /usr/lib/libm.a
```

– Bởi vì thư viện bắt buộc phải có tiếp đầu ngữ lib và có phần mở rộng là .a hoặc .so, trình biên dịch cho phép chúng ta sử dụng tùy chọn `-l` ngắn gọn như sau:

```
$ gcc test.c -o test -lm
```

chúng ta sẽ thấy rằng gcc sẽ mở rộng `-l` thành tiếp đầu ngữ lib và tìm libm.a hoặc libm.so trong thư mục chuẩn để liên kết.

– Mặc dù vậy, không phải lúc nào thư viện của chúng ta cũng phải nằm trong thư viện của Linux. Nếu thư viện của chúng ta nằm ở một thư mục khác, chúng ta có thể chỉ định gcc tìm kiếm trực tiếp với tùy chọn `-L` như sau:

```
$ gcc test.c -o test -L/usr/myproj/lib -ltool
```

– Lệnh trên cho phép liên kết với thư viện libtool.a hoặc libtool.so trong thư mục `/usr/myproj/lib`.

7. Thư viện liên kết trên Linux

– Hình thức đơn giản nhất của thư viện là tập hợp các tập tin .o do trình biên dịch tạo ra ở bước biên dịch với tùy chọn `-c`. Ví dụ

```
$gcc -c helloworld.c
```

trình biên dịch chưa tạo ra tập tin thực thi mà tạo ra tập tin đối tượng `helloworld.o`. Tập tin này chứa các mã máy của chương trình đã được sắp xếp lại. Nếu muốn tạo ra tập tin thực thi, chúng ta gọi trình biên dịch thực hiện bước liên kết:

```
$gcc helloworld.o -o helloworld
```

– Trình biên dịch sẽ gọi tiếp trình liên kết ld tạo ra định dạng tập tin thực thi cuối cùng. Ở đây, nếu chúng ta không sử dụng tùy chọn `-c`, trình biên dịch sẽ thực hiện cả hai bước đồng thời.

a) Thư viện liên kết tĩnh

– Thư viện liên kết tĩnh là các thư viện khi liên kết trình biên dịch sẽ lấy toàn bộ mã thực thi của hàm trong thư viện đưa vào chương trình chính. Chương trình sử dụng thư viện liên kết tĩnh chạy độc lập với thư viện sau khi biên dịch xong. Nhưng khi nâng cấp và sửa đổi, muốn tận dụng những chức năng mới của thư viện thì chúng ta phải biên dịch lại chương trình.

– Ví dụ sử dụng liên kết tĩnh:

```
/* cong.c */
```

```
int cong( int a, int b )
```

```
{
```

```
    return a + b;
```

```
}
```

```
/* nhan.c */
```

```
long nhan( int a, int b )
```

```
{  
  
    return a * b;  
  
}
```

– Thực hiện biên dịch để tạo ra hai tập tin thư viện đối tượng .o \$ gcc -c cong.c nhan.c

– Để một chương trình nào đó gọi được các hàm trong thư viện trên, chúng ta cần tạo một tập tin header .h khai báo các nguyên mẫu hàm để người sử dụng triệu gọi:

```
/* lib.h */
```

```
int cong( int a, int b );
```

```
long nhan( int a, int b );
```

– Cuối cùng, tạo ra chương trình chính program.c triệu gọi hai hàm này.

```
/* program.c */
```

```

#include    <stdio.h>
#include    "lib.h"
int main    ()
{
    int a,    b;

    printf( "Nhập vào a : " );
    scanf(    "%d", &a );
    printf("Nhập vào b        :");
    scanf(    "%d", &b );
    printf( "Tổng %d +        %d = %d\n", a, b, cong( a, b ) );
    printf( "Tích %d *        %d = %d\n", a, b, nhan( a, b ) );
    return    (0);
}

```

– Chúng ta biên dịch và liên kết với chương trình chính như sau: \$ gcc -c program.c

\$ gcc program.o cong.o nhan.o -oprogram

Sau đó thực thi chương trình \$./program

Ở đây .o là các tập tin thư viện đối tượng. Các tập tin thư viện .a là chứa một tập hợp các tập tin .o. Tập tin thư viện .a thực ra là 1 dạng tập tin nén được tạo ra bởi chương trình ar. Chúng ta hãy yêu cầu ar đóng cong.o và nhan.o vào libfoo.a

\$ ar cvr libfoo.a cong.o nhan.o

Sau khi đã có được thư viện libfoo.a, chúng ta liên kết lại với chương trình theo cách sau: \$ gcc program.o -oprogram libfoo.a

Chúng ta có thể sử dụng tùy chọn -l để chỉ định thư viện khi biên dịch thay cho cách trên. Tuy nhiên libfoo.a không nằm trong thư mục thư viện chuẩn, cần phải kết hợp với tùy chọn -L để chỉ định đường dẫn tìm kiếm thư viện trong thư mục hiện hành. Dưới đây là cách biên dịch:

\$ gcc program.c -oprogram -L -lfoo

Chúng ta có thể sử dụng lệnh nm để xem các hàm đã biên dịch sử dụng trong tập tin chương trình, tập tin đối tượng .o hoặc tập tin thư viện .a. Ví dụ:

\$ nm cong.o

b) Thư viện liên kết động

– Khuyết điểm của thư viện liên kết tĩnh là nhúng mã nhị phân kèm theo chương trình khi biên dịch, do đó tốn không gian đĩa và khó nâng cấp. Thư viện liên kết động được dùng để giải quyết vấn đề này. Các hàm trong thư viện liên kết động không trực tiếp đưa vào chương trình lúc biên dịch và liên kết, trình liên kết chỉ lưu thông tin tham chiếu đến các hàm trong thư viện liên kết động. Vào lúc chương trình nhị phân thực thi, Hệ Điều Hành sẽ nạp các chương trình liên kết cần tham chiếu vào bộ nhớ. Như vậy, nhiều chương trình có thể sử dụng chung các hàm trong một thư viện duy nhất.

Tạo thư viện liên kết động:

Khi biên dịch tập tin đối tượng để đưa vào thư viện liên kết động, chúng ta phải thêm tùy chọn –fpic (PIC- Position Independence Code – mã lệnh vị trí độc lập).

Ví dụ: biên dịch lại 2 tập tin cong.c và nhan.c

```
$ gcc -c -fpic cong.c nhan.c
```

Để tạo ra thư viện liên kết động, chúng ta không sử dụng trình ar như với thư viện liên kết tĩnh mà dùng lại gcc với tùy chọn –shared:

```
$ gcc -shared cong.o nhan.o -olibfoo.so
```

Nếu tập tin libfoo.so đã có sẵn trước thì không cần dùng đến tùy chọn –o:

```
$ gcc -shared cong.o nhan.o libfoo.so
```

Bây giờ chúng ta đã có thư viện liên kết động libfoo.so. Biên dịch lại chương trình như sau:

```
$ gcc program.c -oprogram -L. -lfoo
```

Sử dụng thư viện liên kết động:

Khi Hệ Điều Hành nạp chương trình program, nó cần tìm thư viện libfoo.so ở đâu đó trong hệ thống. Ngoài các thư mục chuẩn, Linux còn tìm thư viện liên kết động trong đường dẫn của biến môi trường LD_LIBRARY_PATH. Do libfoo.so đặt trong thư mục hiện hành, không nằm trong các thư mục chuẩn nên ta cần đưa thư mục hiện hành vào biến môi trường LD_LIBRARY_PATH:

```
$ LD_LIBRARY_PATH=.
```

```
$ export LD_LIBRARY_PATH
```

Kiểm tra xem Hệ Điều Hành có thể tìm ra tất cả các thư viện liên kết động mà chương trình sử dụng hay không:

```
$ ldd program
```

rồi chạy chương trình sử dụng thư viện liên kết động này:

```
$/program
```

Một khuyết điểm của việc sử dụng thư viện liên kết động đó là thư viện phải tồn tại trong đường dẫn để Hệ Điều Hành tìm ra khi chương trình được triệu gọi. Nếu không tìm thấy thư viện, Hệ Điều Hành sẽ chấm dứt ngay chương trình cho dù các hàm trong thư viện chưa được sử dụng. Ta có thể chủ động nạp và gọi các hàm trong thư viện liên kết động mà không cần nhờ vào Hệ Điều Hành bằng cách gọi hàm liên kết muộn.

II. Bài thực hành trên lớp

Bài 1. Viết chương trình C in ra màn hình các số nguyên từ 0 đến 9

Bước 1: Mở chương trình soạn thảo: \$vi thuchanh.c

Bước 2: Viết chương trình:

- Khởi đầu vào màn hình vi bạn đang ở chế độ xem (view). Muốn chỉnh sửa nội dung file bạn nhấn phím Insert. Dòng trạng thái cuối màn hình đổi thành --INSERT-- cho biết bạn đang trong chế độ soạn thảo (bạn cũng có thể nhấn phím i hoặc a thay cho phím Insert).
- Nhấn Enter nếu bạn muốn sang dòng mới. Nhấn các phím mũi tên để di chuyển con trỏ và thay đổi nội dung file. Muốn ghi lại nội dung file sau khi soạn thảo, bạn nhấn Esc để trở về chế độ lệnh và nhấn :w. Muốn thoát khỏi vi bạn nhấn :q (hoặc :wq để lưu và thoát).



```
1 //Bai Tap 1
2 #include <stdio.h>
3
4 int main()
5 {
6     int i;
7     for(i=0; i<10; i++)
8         printf("%3d", i);
9     printf("\n");
10    return 0;
11 }
```

Bước 3: Biên dịch chương trình thành tập tin đối tượng: \$gcc -c thuchanh.c

Bước 4: Biên dịch tập tin đối tượng thành tập tin thực thi: \$gcc thuchanh.o -o thuchanh

->Lưu ý: Có thể gom bước 3 và 4 bằng câu lệnh: \$gcc thuchanh.c -o thuchanh

Bước 5: Thực thi chương trình bằng lệnh: `$/thuchanh`



```
Bash
dsl@box:~/thuchanh$ vi thuchanh.c 1
dsl@box:~/thuchanh$
dsl@box:~/thuchanh$ gcc -c thuchanh.c 3
dsl@box:~/thuchanh$
dsl@box:~/thuchanh$ gcc thuchanh.o -o thuchanh 4
dsl@box:~/thuchanh$
dsl@box:~/thuchanh$ ./thuchanh 5
0 1 2 3 4 5 6 7 8 9
dsl@box:~/thuchanh$
```

Bài 2. Viết chương trình cộng và nhân 2 số nguyên sử dụng thư viện liên kết tĩnh:

\$ vi cong.c

```
int cong(int a, int b)
{
    return a + b;
}
```

\$ vi nhan.c

```
int nhan(int a, int b)
{
    return a * b;
}
```

\$ vi program.c

```
#include <stdio.h>

int main()
{
    int a, b;

    printf("\nNhap a:");
    scanf("%d",&a);

    printf("Nhap b:");
    scanf("%d",&b);

    printf("\nTong cua hai so la: %d",cong(a,b));
    printf("\nTich cua hai so la: %d\n",nhan(a,b));
}
```

```
        return 0;
    }
```

```
$ gcc -c cong.c nhan.c
```

```
$ ar cvr libfoo.a cong.o nhan.o
```

```
$ gcc program.c -o program -L. -lfoo
```

```
$ ./program
```

Bài 3. Viết chương trình cộng và nhân 2 số nguyên sử dụng thư viện liên kết động:

```
$ vi cong.c
```

```
int cong(int a, int b)
{
    return a + b;
}
```

```
$ vi nhan.c
```

```
int nhan(int a, int b)
{
    return a * b;
}
```

```
$ vi program.c
```

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int a, b;
```

```
    printf("\nNhap a:");
```

```
    scanf("%d",&a);
```

```
    printf("Nhap b:");
```

```
    scanf("%d",&b);
```

```
    printf("\nTong cua hai so la: %d",cong(a,b)); printf("\nTich  
cua hai so la: %d\n",nhan(a,b)); return 0;
```

```
}
```

```
$ gcc -c -fpic cong.c nhan.c
```

```
$ gcc -shared cong.o nhan.o -o libfoo.so
```

```
$ gcc program.c -o program -L. -lfoo
```

```
$ LD_LIBRARY_PATH=.
```

```
$ export LD_LIBRARY_PATH
```


```
$ ./program
```

4. Bài tập về nhà

Bài tập 1. Viết chương trình nhập, xuất mảng số nguyên(sử dụng thư viện liên kết động).

Bài tập 2. Tạo thư mục **/home/dsl/lib**

- Chép thư viện **libfoo.so** tạo được ở Bài tập 1 vào thư mục vừa tạo.
- Biên dịch và chạy lại chương trình.

Trường: ĐH CNTP TP.HCM Khoa: Công nghệ Thông tin Bộ môn: Mạng máy tính – Truyền thông MH: TH Hệ điều hành MSMH:	BÀI 3. TIẾN TRÌNH	
--	--------------------------	---

D. MỤC TIÊU

Sau khi hoàn thành bài thực hành này, sinh viên sẽ đạt được các kỹ năng sau:

- Hiểu được tiến trình là gì?
- Giám sát và điều khiển các tiến trình

E. DỤNG CỤ - THIẾT BỊ THỰC HÀNH CHO MỘT SINH VIÊN:

STT	Chủng loại – quy cách vật tư	Số lượng	Đơn vị	Ghi chú
1	Computer	1	Bộ	

F. NỘI DUNG THỰC HÀNH

1. Tóm tắt lý thuyết

- Khái niệm tiến trình
- Trạng thái tiến trình
- Giám sát và điều khiển tiến trình.

2. Bài thực hành trên lớp

2.1. Bài tập mẫu

Lấy thông tin trạng thái của các tiến trình: sử dụng câu lệnh ps, pstree, top

PS: #ps <option>

Option:

- f: thể hiện các process dưới dạng tree
- l: thể hiện dưới dạng long list
- w: thể hiện dưới dạng wide output

x: Xem cả các process không gắn với terminal (daemon)

a: process của các user khác

U: user xem process của một user cụ thể

```
root@NTHieu:~  
File Edit View Terminal Go Help  
[root@NTHieu root]# ps l  
F  UID  PID  PPID  PRI  NI   VSZ  RSS  WCHAN  STAT TTY        TIME COMMAND  
0    0  2449   2447   15   0  4324 1408 wait4  S    pts/0        0:00 bash  
0    0  2487   2449   15   0  3920 1048 finish T    pts/0        0:00 top  
0    0  2489   2449   20   0  3120 1172 -      R    pts/0        0:00 ps l  
[root@NTHieu root]#
```

Ý nghĩa các trường:

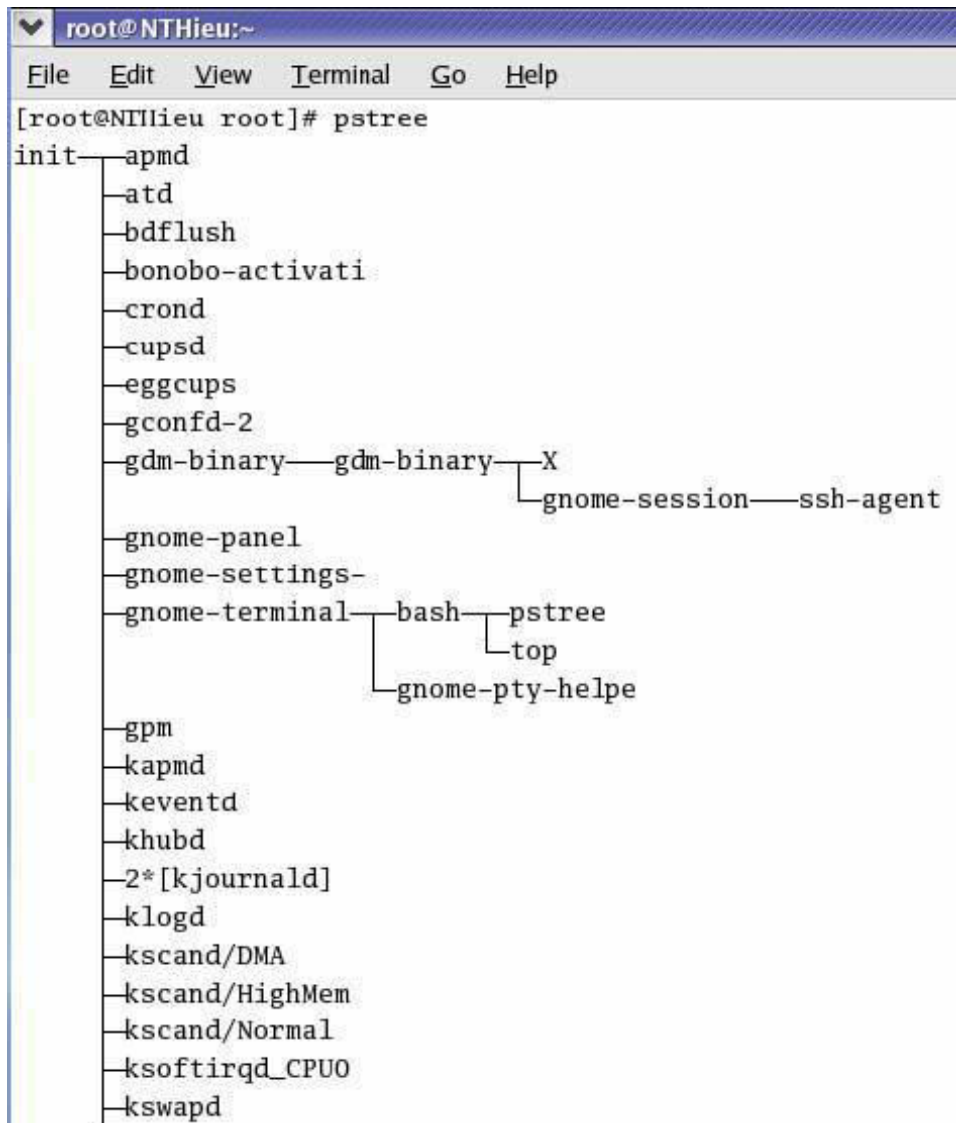
Trường	Giải Thích
USER hoặc UID	Tên của tiến trình
PID	ID (định danh) của tiến trình
%CPU	% CPU sử dụng của tiến trình
%MEM	% bộ nhớ tiến trình sử dụng
SIZE	Kích thước bộ nhớ ảo tiến trình sử dụng

RSS	Kích thước của bộ nhớ thực sử dụng bởi tiến trình
TTY	Vùng làm việc của tiến trình
STAT	Trạng thái của tiến trình
START	Thời gian hay ngày bắt đầu của tiến trình
TIME	Tổng thời gian sử dụng CPU
COMMAND	Câu lệnh được thực hiện
PRI	Mức ưu tiên của tiến trình
FLAGS	Số cờ được kết hợp với tiến trình
PPID	ID của tiến trình cha
WCHAN	Tên của hàm nhân khi tiến trình ngủ được lấy từ file /boot/System.map

Pstree:

Tương tự lệnh ps với tham số -f Tham số

-p in ra màn hình cả process ID



Top: giống lệnh ps nhưng danh sách các process được update liên tục. Các thông số về CPU, RAM cũng được thể hiện và Update. Tham số -d (delay: khoảng thời gian refresh giữa 2 lần), -n (number: chạy n lần và ngưng)

root@NTHieu:~												
File Edit View Terminal Go Help												
08:40:57 up 43 min, 2 users, load average: 0.20, 0.05, 0.01												
57 processes: 52 sleeping, 4 running, 0 zombie, 1 stopped												
CPU states: 30.2% user 7.8% system 0.0% nice 0.0% iowait 62.0% idle												
Mem: 255264k av, 149084k used, 106180k free, 0k shrd, 24396k buff												
131072k actv, 384k in_d, 1524k in_c												
Swap: 0k av, 0k used, 0k free 57408k cached												
PID	USER	PRI	NI	SIZE	RSS	SHARE	STAT	%CPU	%MEM	TIME	CPU	COMMAND
2329	root	16	0	30708	13M	1956	R	34.5	5.4	0:18	0	X
2447	root	15	0	9684	9680	7060	R	2.5	3.7	0:01	0	gnome-terminal
2427	root	15	0	11160	10M	8360	S	0.3	4.3	0:00	0	gnome-panel
2434	root	15	0	6832	6832	5660	S	0.1	2.6	0:00	0	eggccups
2438	root	15	0	13400	13M	8700	R	0.1	5.2	0:01	0	rhn-applet-gui
1	root	15	0	476	476	424	S	0.0	0.1	0:03	0	init
2	root	15	0	0	0	0	SW	0.0	0.0	0:00	0	keventd
3	root	15	0	0	0	0	SW	0.0	0.0	0:00	0	kapmd
4	root	34	19	0	0	0	SWN	0.0	0.0	0:00	0	ksoftirqd_CPU0
9	root	25	0	0	0	0	SW	0.0	0.0	0:00	0	bdfush
5	root	15	0	0	0	0	SW	0.0	0.0	0:00	0	kswapd
6	root	15	0	0	0	0	SW	0.0	0.0	0:00	0	kscand/DMA
7	root	15	0	0	0	0	SW	0.0	0.0	0:00	0	kscand/Normal
8	root	15	0	0	0	0	SW	0.0	0.0	0:00	0	kscand/HighMem
10	root	15	0	0	0	0	SW	0.0	0.0	0:00	0	kupdated
11	root	25	0	0	0	0	SW	0.0	0.0	0:00	0	mdrecoveryd
19	root	15	0	0	0	0	SW	0.0	0.0	0:00	0	kjournald
77	root	25	0	0	0	0	SW	0.0	0.0	0:00	0	khubd
1701	root	15	0	0	0	0	SW	0.0	0.0	0:00	0	kjournald

Gửi tín hiệu cho một tiến trình đang chạy

Lệnh kill:

kill <signal | number> <process id>

Signal	Number	Ý nghĩa
INT	2	Interrupt, được gửi khi ấn phím Ctrl – C
KILL	9	Kill, stop process unconditionally
TERM	15	Terminate, nicely if possible
TSTP	20	Stop executing, ready to continue (tạm dừng)
CONT	18	continue execution, tiếp tục 1 process đã tạm dừng

Thay đổi thông số Priority

Sử dụng lệnh **nice, renice**

Nice: dùng để thay đổi nice number của các process tại thời điểm start time

nice [-n number] [command]

Ví dụ: # nice -n -10 vi /root/data.txt Renice: Thay đổi thông số nice number của các process đã chạy

renice priority PID [[-g] group] [[-u] user]

Ví dụ: # renice -2 203 Set nice number is -2 to PID=203

Can thiệp vào hoạt động

&: Cho một job hoạt động ở background

Ví dụ: # ls -l -R / > /root/list.txt & Ứng dụng ls sẽ chạy nền bên dưới

Ngưng và tạm ngưng Job


Ctrl C: Ngưng job đang thực thi. Sau khi ấn Ctrl C ta có thể dùng câu lệnh jobs để hiển thị trạng thái của các tiến trình đang chạy.

```
[root@NTHieu root]# jobs
[1]-  Stopped                  top
[2]+  Stopped                  top
[root@NTHieu root]#
```

Ctrl Z: Tạm ngưng job đang thực thi. Sau khi ấn Ctrl Z ta có thể dùng 2 câu lệnh: bg: tiếp tục job vừa ngưng ở trạng thái background fg: tiếp tục job vừa ngưng ở trạng thái foreground

2.2. Bài tập luyện tập

1. Xem danh sách các process đang chạy trên hệ thống bằng ps
2. Xem danh sách các process đang chạy trên hệ thống bằng pstree
3. Xem danh sách các process đang chạy trên hệ thống bằng top
4. Lưu các kết quả của lệnh top vào file /root/top.txt
5. Xem tỉ lệ CPU, RAM hệ thống đang sử dụng của từng process đang chạy
6. In thông tin process đang sử dụng nhiều CPU nhất
7. Đếm số process đang thực thi trên máy
8. Đếm số process của user root đang thực thi trên máy
9. Đếm số process “httpd” của user root đang thực thi trên máy
10. Cho biết có bao nhiêu process đang chạy trong hệ thống (dùng wc -l)
11. Mở 2 cửa sổ console (có thể dùng putty). Thực thi cùng lệnh “ls -lR /” với độ ưu tiên lần lượt là -19 và +19, kiểm nghiệm xem lệnh nào sẽ thực thi xong trước
12. Thực thi câu lệnh tìm kiếm tất cả các file có kích thước $\geq 100\text{Kb}$ với độ ưu tiên -5
13. Tăng độ ưu tiên của câu lệnh trên (renice)
14. Ngắt câu lệnh trên

Trường: ĐH CNTP TP.HCM Khoa: Công nghệ Thông tin Bộ môn: Mạng máy tính – Truyền thông MH: TH Hệ điều hành MSMH:	BÀI 4. TIỂU TRÌNH	
--	--------------------------	---

G. MỤC TIÊU

Sau khi hoàn thành bài thực hành này, sinh viên sẽ đạt được các kỹ năng sau:

- Viết được chương trình minh họa thực thi một tiểu trình bằng ngôn ngữ C#.
- Áp dụng được các ngôn ngữ lập trình khác nhau để viết minh họa giải thuật.
- Hình thành tư duy xử lý bài toán lập trình.

H. DỤNG CỤ - THIẾT BỊ THỰC HÀNH CHO MỘT SINH VIÊN:

STT	Chủng loại – quy cách vật tư	Số lượng	Đơn vị	Ghi chú
1	Computer	1	Bộ	

I. NỘI DUNG THỰC HÀNH

3. Tóm tắt lý thuyết

Bài thực hành liên quan đến các vấn đề lý thuyết về tiến trình, cụ thể như sau:

- ▶ Tạo và thực thi thread
- ▶ Truyền tham số cho Thread
- ▶ Property ThreadState và ThreadPriority
- ▶ Các phương thức thông dụng của Thread
- ▶ Foreground và Background Thread
- ▶ Thread Pooling
- ▶ Đồng bộ hóa và locking
- ▶ Deadlock

4. Bài thực hành trên lớp

4.1. Bài tập mẫu 1

Yêu cầu: Viết chương trình C# minh họa việc tạo và thực thi một tiểu trình

Mô tả vấn đề: Phân biệt được kết quả trả về của việc thực thi trên một tiểu trình và thực thi trên 2 tiểu trình

Chương trình mẫu:

```
class Program
{
    static void Main()
    {
        Thread t = new Thread(new ThreadStart(MethodA));
        t.Start();
        MethodB();
    }
    static void MethodA()
    {
        for (int i = 0; i < 100; i++)
            Console.Write("0");
    }
    static void MethodB()
    {
        for (int i = 0; i < 100; i++)
            Console.Write("1");
    }
}
```

Kết quả chạy chương trình:

[illegible]

4.2. Bài tập mẫu 2

Yêu cầu: Viết chương trình C# minh họa việc truyền tham số cho Thread

Mô tả vấn đề: Trong trường hợp bạn muốn truyền tham số cho thread, ta sử dụng ParameterizedThreadStart thay thế cho ThreadStart

Chương trình mẫu:

```
namespace ThreadExample
{
    class Student
    {
        public string Name { get; set; }
        public DateTime BirthDay { get; set; }
    }

    class Program
    {
        static void Main()
        {
            Thread t1 = new Thread(Print);

            t1.Start(new Student() { Name = "ABC", BirthDay = new DateTime(1980,
10, 20) });

            Console.ReadKey();
        }

        static void Print(object obj)
        {
            Student st = (Student)obj;
            Console.WriteLine(st.Name + "\t" + st.BirthDay.ToShortDateString());
        }
    }
}
```

Kết quả chạy chương trình:

ABC 20/10/1980

4.3. Bài tập mẫu 3

Yêu cầu: Viết chương trình C# minh họa việc gọi phương thức Sleep() của Thread

Mô tả vấn đề: Ta sử dụng phương thức Sleep để kiểm tra thứ tự thực hiện giữa 2 tiến trình

Chương trình mẫu:

```
class Program
{
    static void Main()
    {
        Thread t = new Thread(MethodA);
        t.Start();
        MethodB();
    }

    static void MethodA()
    {
        Thread.Sleep(500); // sleep for 500 milliseconds
        for (int i = 0; i < 100; i++)
            Console.Write("0");
    }

    static void MethodB()
    {
        for (int i = 0; i < 100; i++)
            Console.Write("1");
    }
}
```

Kết quả chạy chương trình:

[illegible]

4.5. Bài tập mẫu 5

Yêu cầu: Viết chương trình C# minh họa việc thực thi Foreground và Background Thread

Mô tả vấn đề: Viết đoạn code mô tả việc sử dụng forceground và backgroud Thread. Phân biệt forceground và backgroud Thread trong lập trình

Chương trình mẫu:

```
static void Main(string[] args)
{
    Thread t1 = new Thread(() =>
    {
        Thread.Sleep(1000);
        Console.WriteLine("Thread t1 started");
    });
    // t1.IsBackground = true;
    t1.Start();
    Console.WriteLine("Main thread ending...");
}
```

Kết quả chạy chương trình:

Main thread ending...

Thread t1 started

Lưu ý: Bây giờ bạn uncomment dòng `t1.IsBackground = true` và chạy lại, kết quả sẽ chỉ xuất ra một dòng sau: `Main thread ending...`

4.6. Bài tập mẫu 6

Yêu cầu: Viết chương trình C# minh họa việc thực thi Thread Pooling

Mô tả vấn đề: Viết đoạn code mô tả việc sử dụng Thread Pooling. Xác định vai trò của thread pooling trong lập trình

Chương trình mẫu:

```
class ThreadPooling
{
    static void Main()
    {
        ThreadPool.QueueUserWorkItem(ThreadProc);
        ThreadPool.QueueUserWorkItem(ThreadProc, 123);
    }

    static void ThreadProc(object data)
    {
        for (int i = 0; i < 10; i++)
        {
            Console.WriteLine("Thread callback: " + data);
            Thread.Sleep(500);
        }
    }
}
```

Kết quả chạy chương trình:

Quan sát (process:****), với **** được tạo ngẫu nhiên

Lưu ý: Bạn có thể thay đổi số thread lớn nhất mà thread pool tạo ra bằng cách sử dụng phương thức ThreadPool.SetMaxThreads(). Trong mỗi phiên bản .Net giá trị mặc định này không giống nhau, ví dụ phiên bản .Net 2.0 thì giá trị này là 25, trong .Net 3.5 là 250. Bạn có thể kiểm tra điều này bằng cách sử dụng phương thức ThreadPool.GetMaxThreads().

4.7. Bài tập mẫu 7

Yêu cầu: Viết chương trình C# minh họa việc đồng bộ hóa và locking tiểu trình

Mô tả vấn đề: Viết đoạn code mô tả việc đồng bộ hóa và locking tiểu trình.

Chương trình mẫu:

```
class ThreadLocking
{
    static int amount = 0;

    static void Main()
    {
        Thread t1 = new Thread(IncreaseAmount);
        Thread t2 = new Thread(DecreaseAmount);

        t1.Start();
        t2.Start();
    }

    static void IncreaseAmount()
    {
        for (int i = 0; i < 100; i++)
        {
            amount++;

            if (amount > 0)
            {
                Thread.Sleep(1);
                Console.Write(amount + "\t");
            }
        }
    }

    static void DecreaseAmount()
    {
        for (int i = 0; i < 100; i++)
        {
            amount--;
        }
    }
}
```

Kết quả chạy chương trình:

1 -98

Lưu ý: Phương thức IncreaseAmount() sẽ in ra màn hình giá trị của biến amount chỉ khi biến này có giá trị lớn hơn 0

4.8. Bài tập mẫu 8

Yêu cầu: Viết chương trình C# minh họa hiện tượng deadlock của tiểu trình

Mô tả vấn đề: Viết đoạn code mô tả hiện tượng deadlock của tiểu trình.

Chương trình mẫu:

```
class ThreadDeadlock
{
    static object syncObj1 = new object();
    static object syncObj2 = new object();

    static void Main()
    {
        Thread t1 = new Thread(Foo);
        Thread t2 = new Thread(Bar);

        t1.Start();
        t2.Start();
    }

    static void Foo()
    {
        Console.WriteLine("Inside Foo method");
        lock (syncObj1)
        {
            Console.WriteLine("Foo: lock(syncObj1)");
            Thread.Sleep(100);
            lock (syncObj2)
            {
                Console.WriteLine("Foo: lock(syncObj2)");
            }
        }
    }

    static void Bar()
    {
        Console.WriteLine("Inside Bar method");
        lock (syncObj2)
        {
            Console.WriteLine("Bar: lock(syncObj2)");
            Thread.Sleep(100);
            lock (syncObj1)
            {
                Console.WriteLine("Bar: lock(syncObj1)");
            }
        }
    }
}
```

Kết quả chạy chương trình:

Inside Foo method
Foo: lock(syncObj1)
Inside Bar method
Bar: lock(syncObj2)

Lưu ý: Với mỗi phương thức, ta sử dụng Thread.Sleep(100) để thread chứa phương thức kia có thời gian thực thi trước khi phương thức này kết thúc

4.9. Bài tập luyện tập

Bài tập 1. Truyền vào một số n, thực hiện hiển thị chuỗi Fibonacci đến n

Bài tập 2. Tạo 5 thread, mỗi thread thực hiện xuất ra một chuỗi “hello”

Bài tập 3. Sử dụng Thread để tính tổng các phần tử trong list

Chương trình được xây dựng để tính tổng cho một list có n phần tử, có giá trị ngẫu nhiên từ 0 -10

Yêu cầu tạo ra Thread để tính tổng trong một list con (1 phần trong list lớn)

```

Nhập số phần tử:      40
List: [7, 3, 5, 4, 7, 4, 6, 2, 1, 7, 7, 7, 4, 7, 0, 7, 2, 2, 4, 5, 6, 0, 2, 5, 8, 0, 5, 4, 7, 3, 4, 1, 0, 0, 9, 6, 5, 0, 3, 9]
Nhập vào số thread:  3
Thread 1 : 7; 3; 5; 4; 7; 4; 6; 2; 1; 7; 7; 7; 4;
Thread 2 : 7; 0; 7; 2; 2; 4; 5; 6; 0; 2; 5; 8; 0;
Thread 3 : 5; 4; 7; 3; 4; 1; 0; 0; 9; 6; 5; 0; 3; 9;
Tổng list= 168
```


5. Bài thực hành ở nhà

Hãy thực hiện bài tập sau bằng C#. Yêu cầu: Nộp lại source code vào thư mục trên kênh học tập Google Classroom.

Bài tập 1. Tìm giá trị lớn nhất trong list

Sử dụng Thread để tìm giá trị lớn nhất trong list

- ▶ Chương trình được xây dựng để tìm giá trị lớn nhất trong một list có n phần tử, có giá trị ngẫu nhiên từ 0 -100
- ▶ Yêu cầu tạo ra Thread để tìm max trong một list con (1 phần trong list lớn)
- ▶ Hãy tạo ra một số Thread để thực hiện việc tìm max này

Trường: ĐH CNTP TP.HCM Khoa: Công nghệ Thông tin Bộ môn: Mạng máy tính – Truyền thông MH: TH Hệ điều hành MSMH:	BÀI 5. GIẢI THUẬT ĐỊNH THỜI	
--	------------------------------------	---

J. MỤC TIÊU

Sau khi hoàn thành bài thực hành này, sinh viên sẽ đạt được các kỹ năng sau:

- Viết được chương trình minh họa việc thực thi của giải thuật định thời CPU như: FCFS, SJF, RR, Priority bằng ngôn ngữ C.
- Áp dụng được các ngôn ngữ lập trình khác nhau để viết minh họa giải thuật.
- Hình thành tư duy xử lý bài toán lập trình.

K. DỤNG CỤ - THIẾT BỊ THỰC HÀNH CHO MỘT SINH VIÊN:

STT	Chủng loại – quy cách vật tư	Số lượng	Đơn vị	Ghi chú
1	Computer	1	Bộ	

L. NỘI DUNG THỰC HÀNH

6. Tóm tắt lý thuyết

Bài thực hành liên quan đến các vấn đề lý thuyết về định thời CPU, cụ thể như sau:

- Giải thuật định thời FCFS (Đến trước được phục vụ trước)
- Giải thuật định thời SJF (Tác vụ ngắn nhất đầu tiên)
- Giải thuật RR (Xoay vòng)
- Giải thuật Priority (Độ ưu tiên)

7. Bài thực hành trên lớp

7.1. Bài tập mẫu

Yêu cầu: Viết chương trình C minh họa việc thực thi giải thuật FCFS?

Mô tả vấn đề: Trong giải thuật này, bộ lập lịch CPU sẽ thực thi theo cơ chế độc quyền, tiến trình đến hàng đợi đầu tiên sẽ được cấp cho CPU để hoàn thành yêu cầu của nó, chỉ khi tiến trình kết thúc CPU mới thực hiện trên tiến trình khác.

Các bước thực hiện Giải thuật FCFS:

- Bước 1: Khai báo cấu trúc tiến trình với các biến bên trong gồm mã tiến trình, thời gian chờ, thời gian xử lý của tiến trình, tổng thời gian.
- Bước 2: Khai báo biến trong chương trình chính: i, j, n là số nguyên, tổng thời gian chờ, tổng thời gian hoàn tất, thời gian chờ trung bình, thời gian hoàn tất trung bình.
- Bước 3: Khởi tạo thời gian chờ cho tiến trình đầu tiên = 0 và tổng thời gian cho tiến trình đầu tiên.
- Bước 4: Tính tổng thời gian và thời gian xử lý cho các tiến trình còn lại.
- Bước 5: Thời gian chờ của một tiến trình là tổng thời gian của tiến trình trước đó.
- Bước 6: Tổng thời gian của một tiến trình = thời gian chờ + thời gian xử lý
- Bước 7: Tổng thời gian chờ được tính bằng cách cộng thêm thời gian chờ của các tiến trình đã chạy.
- Bước 8: Tổng thời gian hoàn tất được tính bằng cách cộng tất cả thời gian hoàn tất của các tiến trình đã chạy.
- Bước 9: Tính thời gian chờ trung bình.
- Bước 10: Tính thời gian hoàn tất trung bình.
- Bước 11: Chạy chương trình kiểm tra kết quả.

Chương trình mẫu:

```

1  /*****
2  CHƯƠNG TRÌNH MINH HỌA GIẢI THUẬT ĐỊNH THỜI CPU
3  GIẢI THUẬT FCFS
4  MÔN: THỰC HÀNH HỆ ĐIỀU HÀNH
5  BỘ MÔN: MẠNG MÁY TÍNH & TRUYỀN THÔNG
6  KHOA: CNTT - ĐH CNTP TPHCM
7  *****/
8
9  #include<stdio.h>
10 struct process
11 {
12     int ma_tientrinh, tgcho, tgxuly, tong_tgian;
13 } p[20];
14 main ()
15 {
16     int i, n, j, tong_tgcho = 0, tong_tghoantat = 0, tg_cho_tb, tg_hoantat_tb;
17     printf ("Nhap so tien trinh: ");
18     scanf ("%d", &n);
19     for (i = 1; i <= n; i++)
20     {
21         printf ("Nhap ma dinh danh tien trinh: ");
22         scanf ("%d", &p[i].ma_tientrinh);
23         printf ("Nhap thoi gian xu ly cua tien trinh: ");
24         scanf ("%d", &p[i].tgxuly);
25     }
26     p[1].tgcho = 0;
27     p[1].tong_tgian = p[1].tgxuly;
28     for (i = 2; i <= n; i++)
29     {
30         for (j = 1; j < i; j++)
31         {
32             p[i].tgcho = p[i].tgcho + p[j].tgxuly;
33         }
34         tong_tgcho = tong_tgcho + p[i].tgcho;
35         p[i].tong_tgian = p[i].tgcho + p[i].tgxuly;
36         tong_tghoantat = tong_tghoantat + p[i].tong_tgian;
37     }
38     tg_hoantat_tb = tong_tghoantat / n;
39     tg_cho_tb = tong_tgcho / n;
40     printf ("TT\tTG-XuLy\tTG-Cho\tTong");
41     for (i = 1; i <= n; i++)
42     {
43         printf ("\n%d\t%d\t%d\t%d\n", p[i].ma_tientrinh, p[i].tgxuly,
44             p[i].tgcho, p[i].tong_tgian);
45     }
46     printf ("Thoi gian cho trung binh: %d\n", tg_cho_tb);
47     printf ("Thoi gian hoan tat trung binh: %d\n", tg_hoantat_tb);
48 }

```

Kết quả chạy chương trình:

```
Nhap so tien trinh: 3
Nhap ma dinh danh tien trinh: 1
Nhap thoi gian xu ly cua tien trinh: 24
Nhap ma dinh danh tien trinh: 2
Nhap thoi gian xu ly cua tien trinh: 3
Nhap ma dinh danh tien trinh: 3
Nhap thoi gian xu ly cua tien trinh: 3
TT      TG-XuLy  TG-Cho  Tong
1        24      0      24
2         3     24     27
3         3     27     30
Thoi gian cho trung binh: 17
Thoi gian hoan tat trung binh: 19
```

7.2. Bài tập luyện tập

- Bài tập 1. Viết chương trình C minh họa việc thực thi giải thuật SJF?
- Bài tập 2. Viết chương trình C minh họa việc thực thi giải thuật RR?
- Bài tập 3. Viết chương trình C minh họa việc thực thi giải thuật Priority?

8. Bài thực hành ở nhà

Hãy thực hiện lại các bài tập sau bằng Windows Form. Yêu cầu: Nộp lại source code vào thư mục trên kênh học tập Google Classroom.

- Bài tập 1. Viết chương trình minh họa việc thực thi giải thuật SJF?
- Bài tập 2. Viết chương trình minh họa việc thực thi giải thuật RR?
- Bài tập 3. Viết chương trình minh họa việc thực thi giải thuật Priority?

Ví dụ giao diện Giải thuật FCFS

Giao Diện Giải Thuật FCFS

NHẬP THÔNG TIN

Tiến trình

Nhập lại

Thời gian thực thi

Thời gian đến

Thêm

Tiến trình	Thời gian thực thi	Thời gian đến

FCFS

RESET

Thời gian chờ trung bình

Thời gian xử lý trung bình

Trường: ĐH CNTP TP.HCM Khoa: Công nghệ Thông tin Bộ môn: Mạng máy tính – Truyền thông MH: TH Hệ điều hành MSMH:	<h2 style="text-align: center;">BÀI 6. ĐỒNG BỘ HÓA TIỀN TRÌNH</h2>	
---	--	--

M.MỤC TIÊU

Sau khi hoàn thành bài thực hành này, sinh viên sẽ đạt được các kỹ năng sau:

- Viết được chương trình minh họa việc thực thi các bài toán đồng bộ hóa kinh điển như: Producer-Consumer, Reader-Writer, Dining Philosopher bằng ngôn ngữ C, với các giải pháp đồng bộ: Semaphore, Monitor
- Áp dụng được các ngôn ngữ lập trình khác nhau để viết minh họa giải thuật.
- Hình thành tư duy xử lý bài toán lập trình.

N. DỤNG CỤ - THIẾT BỊ THỰC HÀNH CHO MỘT SINH VIÊN:

STT	Chủng loại – quy cách vật tư	Số lượng	Đơn vị	Ghi chú
1	Computer	1	Bộ	

O. NỘI DUNG THỰC HÀNH

9. Tóm tắt lý thuyết

Bài thực hành liên quan đến các vấn đề lý thuyết về đồng bộ hóa tiến trình, cụ thể như sau:

- Đồng bộ hóa bằng giải pháp Semaphore
- Đồng bộ hóa bằng giải pháp Monitor

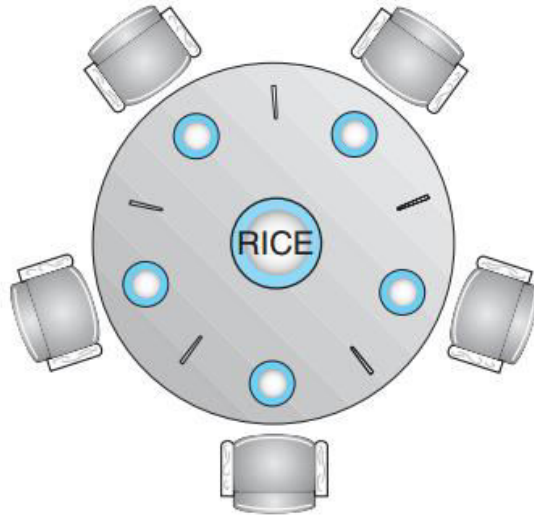
10. Bài thực hành trên lớp

10.1. Bài tập mẫu

Yêu cầu: Viết chương trình C minh họa đồng bộ hóa tiến trình bằng Semaphore thông qua bài toán Bữa ăn tối của các triết gia?

Mô tả vấn đề:

- Thuật ngữ: the dining philosophers problem
- Có 5 triết gia, 5 chiếc đũa, 5 bát cơm và một âu cơm bố trí như hình vẽ
- Đây là bài toán cổ điển và là ví dụ minh họa cho một lớp nhiều bài toán tranh đoạt điều khiển: Nhiều tiến trình khai thác nhiều tài nguyên chung



Chương trình mẫu:

```
1  /*****
2  CHƯƠNG TRÌNH MINH HỌA ĐỒNG BỘ HÓA TIẾN TRÌNH SỬ DỤNG SEMAPHORES
3  BÀI TOÁN BỮA ĂN TỐI CỦA CÁC TRIẾT GIA
4  MÔN: THỰC HÀNH HỆ ĐIỀU HÀNH
5  BỘ MÔN: MẠNG MÁY TÍNH & TRUYỀN THÔNG
6  KHOA: CNTT - ĐH CNTP
7  *****/
8
9  #include <pthread.h>
10 #include <semaphore.h>
11 #include <stdio.h>
12
13 #define N 5
14 #define SUYNGHI 2
15 #define DOI 1
16 #define AN 0
17 #define TRAI (phnum + 4) % N
18 #define PHAI (phnum + 1) % N
19
20 int state[N];
21 int phil[N] = { 0, 1, 2, 3, 4 }; // 5 triết tra
22
23 sem_t mutex;
24 sem_t S[N];
25
26 void
27 test (int phnum)
28 {
29     if (state[phnum] == DOI
30         && state[TRAI] != AN && state[PHAI] != AN)
31     {
32         // Trạng thái ăn
33         state[phnum] = AN;
34
35         sleep (2);
36
37         printf ("Triết gia %d lấy đĩa %d và %d\n",
38             phnum + 1, TRAI + 1, phnum + 1);
39
40         printf ("Triết gia %d ăn\n", phnum + 1);
41
42         sem_post (&S[phnum]);
43     }
44 }
45
46 // Cầm đĩa
47 void
48 take_fork (int phnum)
49 {
50 }
```

```

51
52     sem_wait (&mutex);
53
54     // trạng thái đói
55     state[phnum] = DOI;
56
57     printf ("Triết gia %d đói\n", phnum + 1);
58
59     // ăn nếu người bên cạnh không ăn
60     test (phnum);
61
62     sem_post (&mutex);
63
64     // Nếu không thể ăn, đợi để được báo hiệu ăn
65     sem_wait (&S[phnum]);
66
67     sleep (1);
68 }
69
70 // đặt đĩa xuống
71 void
72 put_fork (int phnum)
73 {
74
75     sem_wait (&mutex);
76
77     // trạng thái suy nghĩ
78     state[phnum] = SUYNGHI;
79
80     printf ("Triết gia %d đặt đĩa %d và %d xuống\n",
81            phnum + 1, TRAI + 1, phnum + 1);
82     printf ("Triết gia %d suy nghĩ\n", phnum + 1);
83
84     test (TRAI);
85     test (PHAI);
86
87     sem_post (&mutex);
88 }
89
90 void *
91 philospher (void *num)
92 {
93
94     while (1)
95     {
96
97         int *i = num;
98
99         sleep (1);
100

```

```

101     take_fork (*i);
102
103     sleep (0);
104
105     put_fork (*i);
106 }
107 }
108
109 int
110 main ()
111 {
112
113     int i;
114     pthread_t thread_id[N];
115
116     // Khởi tạo semaphores
117     sem_init (&mutex, 0, 1);
118
119     for (i = 0; i < N; i++)
120
121         sem_init (&S[i], 0, 0);
122
123     for (i = 0; i < N; i++)
124     {
125
126         // Tạo các tiến trình "triết gia"
127         pthread_create (&thread_id[i], NULL, philosopher, &phil[i]);
128
129         printf ("Triết gia %d suy nghĩ\n", i + 1);
130     }
131
132     for (i = 0; i < N; i++)
133
134         pthread_join (thread_id[i], NULL);
135 }
136

```

Kết quả chạy chương trình:

```
Triết gia 1 suy nghĩ
Triết gia 2 suy nghĩ
Triết gia 3 suy nghĩ
Triết gia 4 suy nghĩ
Triết gia 5 suy nghĩ
Triết gia 1 đói
Triết gia 2 đói
Triết gia 4 đói
Triết gia 5 đói
Triết gia 5 lấy đĩa 4 và 5
Triết gia 5 ăn
Triết gia 3 đói
Triết gia 3 lấy đĩa 2 và 3
Triết gia 3 ăn
Triết gia 5 đặt đĩa 4 và 5 xuống
Triết gia 5 suy nghĩ
Triết gia 1 lấy đĩa 5 và 1
Triết gia 1 ăn
Triết gia 3 đặt đĩa 2 và 3 xuống
Triết gia 3 suy nghĩ
Triết gia 4 lấy đĩa 3 và 4
Triết gia 4 ăn
Triết gia 5 đói
Triết gia 1 đặt đĩa 5 và 1 xuống
```

10.2. Bài tập luyện tập

Bài tập 1. Viết chương trình C minh họa đồng bộ hóa tiến trình bằng Semaphore thông qua bài toán Producer-Consumer?

Bài tập 2. Viết chương trình C minh họa đồng bộ hóa tiến trình bằng Semaphore thông qua bài toán Reader-Writer?

Bài tập 3. Viết chương trình C minh họa đồng bộ hóa tiến trình bằng Monitor thông qua bài toán Bữa ăn tối của các triết gia?

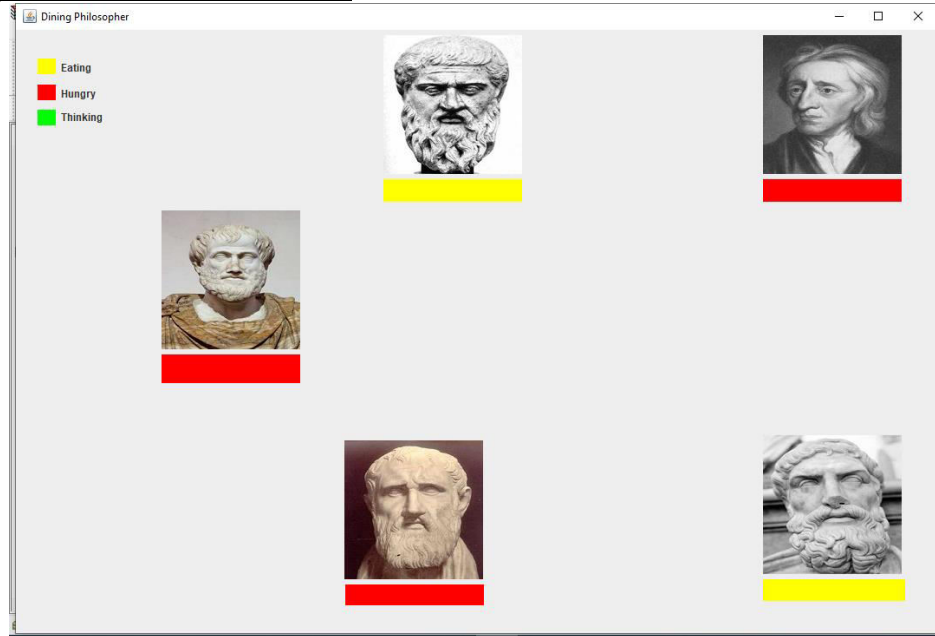
11. Bài thực hành ở nhà


Hãy thực hiện lại các bài tập sau bằng Windows Form. Yêu cầu: Nộp lại source code vào thư mục trên kênh học tập Google Classroom.

- Bài tập 1. Viết chương trình C minh họa đồng bộ hóa tiến trình bằng Semaphore thông qua bài toán Producer-Consumer?
- Bài tập 2. Viết chương trình C minh họa đồng bộ hóa tiến trình bằng Semaphore thông qua bài toán Reader-Writer?

- Bài tập 3. **Viết chương trình C minh họa đồng bộ hóa tiến trình bằng Semaphore thông qua bài toán Bữa ăn tối của các triết gia?**

Ví dụ giao diện Giải thuật FCFS



Trường: ĐH CNTP TP.HCM Khoa: Công nghệ Thông tin Bộ môn: Mạng máy tính – Truyền thông MH: TH Hệ điều hành MSMH:	BÀI 7. TẮC NGHẼN	
---	-------------------------	---

P. MỤC TIÊU

Sau khi hoàn thành bài thực hành này, sinh viên sẽ đạt được các kỹ năng sau:

- Viết được chương trình minh họa việc thực thi của các giải thuật quản lý tắc nghẽn như: Tránh tắc nghẽn, dò tìm tắc nghẽn, ngăn chặn tắc nghẽn bằng ngôn ngữ C.
- Áp dụng được các ngôn ngữ lập trình khác nhau để viết minh họa giải thuật.
- Hình thành tư duy xử lý bài toán lập trình.

Q. DỤNG CỤ - THIẾT BỊ THỰC HÀNH CHO MỘT SINH VIÊN:

STT	Chủng loại – quy cách vật tư	Số lượng	Đơn vị	Ghi chú
1	Computer	1	Bộ	

R. NỘI DUNG THỰC HÀNH

12. Tóm tắt lý thuyết

Bài thực hành liên quan đến các vấn đề lý thuyết về giải quyết tắc nghẽn trong hệ điều hành, cụ thể như sau:

- Giải thuật phát hiện tắc nghẽn
- Giải thuật tránh tắc nghẽn
- Giải thuật ngăn chặn tắc nghẽn

13. Bài thực hành trên lớp

13.1. Bài tập mẫu

Yêu cầu: Viết chương trình C minh họa việc thực thi giải thuật Banker tránh tắc nghẽn?

Chương trình mẫu:

```
1  /*
2  CHƯƠNG TRÌNH MINH HỌA GIẢI THUẬT BANKER
3  MÔN: THỰC HÀNH HỆ ĐIỀU HÀNH
4  BỘ MÔN: MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG
5  KHOA: CNTT - ĐH CNTP
6  */
7  #include <stdio.h>
8  int main()
9  {
10     // P0, P1, P2, P3, P4 Tên của các tiến trình minh họa
11
12     int n, m, i, j, k;
13     n = 5; // Số tiến trình
14     m = 3; // Số tài nguyên
15     int alloc[5][3] = { { 0, 1, 0 }, // P0 // Ma trận Allocation
16                        { 2, 0, 0 }, // P1
17                        { 3, 0, 2 }, // P2
18                        { 2, 1, 1 }, // P3
19                        { 0, 0, 2 } }; // P4
20
21     int max[5][3] = { { 7, 5, 3 }, // P0 // Ma trận MAX
22                     { 3, 2, 2 }, // P1
23                     { 9, 0, 2 }, // P2
24                     { 2, 2, 2 }, // P3
25                     { 4, 3, 3 } }; // P4
26
27     int avail[3] = { 3, 3, 2 }; // Tài nguyên sẵn hiện tại có của hệ thống
28
29     int f[n], ans[n], ind = 0;
30     for (k = 0; k < n; k++) {
31         f[k] = 0;
32     }
33     int need[n][m];
34     for (i = 0; i < n; i++) {
35         for (j = 0; j < m; j++)
36             need[i][j] = max[i][j] - alloc[i][j];
37     }
38     int y = 0;
39     for (k = 0; k < 5; k++) {
40         for (i = 0; i < n; i++) {
41             if (f[i] == 0) {
42
43                 int flag = 0;
44                 for (j = 0; j < m; j++) {
45                     if (need[i][j] > avail[j]){
46                         flag = 1;
47                         break;
48                     }
49                 }
50
51                 if (flag == 0) {
52                     ans[ind++] = i;
53                     for (y = 0; y < m; y++)
54                         avail[y] += alloc[i][y];
55                     f[i] = 1;
56                 }
57             }
58         }
59     }
60
61     printf("Chuỗi an toàn là:\n");
62     for (i = 0; i < n - 1; i++)
63         printf(" %d ->", ans[i]);
64     printf(" %d", ans[n - 1]);
65
66     return (0);
67
68 }
69
70
```


Kết quả chạy chương trình:

Chuỗi an toàn là: P1 -> P3 -> P4 -> P0 -> P2

13.2. Bài tập luyện tập

Bài tập 1. Viết chương trình C minh họa việc thực thi giải thuật Banker dò tìm tắc nghẽn?

Bài tập 2. Viết chương trình C minh họa việc thực thi giải thuật Banker ngăn chặn tắc nghẽn?

14. Bài thực hành ở nhà

Hãy thực hiện lại các bài tập sau bằng Windows Form. Yêu cầu: Nộp lại source code vào thư mục trên kênh học tập Google Classroom.

- Bài tập 1. Viết chương trình C minh họa việc thực thi giải thuật Banker?

Ví dụ giao diện Giải thuật

The screenshot shows the 'bankersAlgo' application window. It contains several input fields and tables for simulating the Banker's Algorithm.

Allocation Table:

Process	A	B	C
P0	0	1	0
P1	2	0	0
P2	3	0	2
P3	2	1	1
P4	0	0	2

Max Allocation Table:

	A	B	C
7	5	3	
3	2	2	
9	0	2	
2	2	2	
4	3	3	

Need Matrix:

	A	B	C
7	4	3	
1	2	2	
6	0	0	
0	1	1	
4	3	1	


Available Matrix:

	A	B	C
5	3	2	
7	4	3	
7	4	5	
7	5	5	
10	5	7	

Safe Sequence:

P1 Allocated
P3 Allocated
P4 Allocated
P0 Allocated
P2 Allocated

Safely allocated

Trường: ĐH CNTP TP.HCM Khoa: Công nghệ Thông tin Bộ môn: Mạng máy tính – Truyền thông MH: TH Hệ điều hành MSMH:	BÀI 8. QUẢN LÝ BỘ NHỚ	
--	------------------------------	---

S. MỤC TIÊU

Viết được chương trình minh họa trên ngôn ngữ C trong việc mô phỏng các kỹ thuật phân phối bộ nhớ liên kế, quản lý và chia sẻ bộ nhớ giữa các process:

- Kỹ thuật phân phối bộ nhớ:
 - Không phù hợp nhất (Worst-fit)
 - Phù hợp nhất (Best-fit)
 - Phù hợp đầu tiên (First-fit)
 - Phân phối cấp phát bộ nhớ trong chia bộ nhớ thành nhiều phân vùng có kích thước cố định
 - Tiến trình kết thúc, phần vùng sẽ cấp cho tiến trình khác.
 - HDH cho biết phần bộ nhớ trống và phần bộ nhớ đã sử dụng
- Phân bổ và quản lý bộ nhớ
 - Phân bổ bộ nhớ
 - Sử dụng thư viện trong tạo, quản lý bộ nhớ: malloc(), calloc(), free() and realloc()
 - Sử dụng mảng trong phân bổ bộ nhớ
- Bộ nhớ chia sẻ (Share memory)
 - Memory page structure: Struct page, Permission, Flag
 - Library support: shm API, mmap, mmap API
 - Sequence diagram to share memory via mmap API: create file fd, set memory size, map shared mem to process, unmap shared mem, remove the shared memory
 - Example code
- Áp dụng được các ngôn ngữ lập trình khác nhau để viết minh họa giải thuật.

T. DỤNG CỤ - THIẾT BỊ THỰC HÀNH CHO MỘT SINH VIÊN:

STT	Chủng loại – quy cách vật tư	Số lượng	Đơn vị	Ghi chú

1	Computer	1	Bộ	
---	----------	---	----	--

U. NỘI DUNG THỰC HÀNH

Bài thực hành liên quan đến các vấn đề lý thuyết về phân phối bộ cụ thể như sau:

1. Bài tập mẫu 1: Phân phối bộ nhớ

1.1. Không phù hợp nhất (Worst-fit)

- Code chương trình

```
#include<stdio.h>
#include<conio.h>
#define max 25
void main()
{
    int
        frag[max], b[max], f[max], i, j, nb, nf, t
        emp; static int bf[max], ff[max];

    clrscr();
    printf("\n\tMemory Management Scheme - First Fit");
    printf("\nEnter the number of blocks:");
    scanf("%d", &nb);
    printf("Enter the number of files:");
    scanf("%d", &nf);
    printf("\nEnter the size of the blocks:-\n");
    for (i = 1; i <= nb; i++)
    {
        printf("Block %d:", i);
        scanf("%d", &b[i]);
    }
    printf("Enter the size of the files :-\n");
    for (i = 1; i <= nf; i++)
    {
        printf("File %d:", i);
        scanf("%d", &f[i]);
    }
    for (i = 1; i <= nf; i++)
    {
        for (j = 1; j <= nb; j++)
        {
            if (bf[j] != 1)
            {
                temp = b[j] - f[i];
                if (temp >= 0)
                {
                    ff[i] = j;
                    break;
                }
            }
        }
        frag[i] = temp;
        bf[ff[i]] = 1;
    }
    printf("\nFile_no:\tFile_size :\tBlock_no:\tBlock_size:\tFragement");
    for (i = 1; i <= nf; i++)
        printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d", i, f[i], ff[i], b[ff[i]], frag[i]);
    getch();
}
```

- Dữ liệu đầu vào:

```

Enter the number of blocks : 3
Enter the number of files : 2
Enter the size of the blocks : -
Block 1 : 5
Block 2 : 2
Block 3 : 7
Enter the size of the files : -
File 1 : 1
File 2 : 4

```

- Kết quả chạy chương trình:

File No	File Size	Block No	Block Size	Fragment
1	1	1	5	4
2	4	3	7	3

1.2. Phù hợp nhất (Best-fit)

- Code chương trình

```

#include<stdio.h>
#include<conio.h>
#define max 25
void main()
{
    int frag[max], b[max], f[max], i, j, nb, nf, temp, lowest = 10000;
    static int bf[max], ff[max];
    clrscr();
    printf("\nEnter the number of blocks:");
    scanf("%d", &nb);
    printf("Enter the number of files:");
    scanf("%d", &nf);
    printf("\nEnter the size of the blocks:-\n");
    for (i = 1; i <= nb; i++)
        printf("Block %d:", i);
    scanf("%d", &b[i]);
    printf("Enter the size of the files :-\n");
    for (i = 1; i <= nf; i++)
    {
        printf("File %d:", i);
        scanf("%d", &f[i]);
    }
    for (i = 1; i <= nf; i++)
    {
        for (j = 1; j <= nb; j++)
        {
            if (bf[j] != 1)
            {
                temp = b[j] - f[i];
                if (temp >= 0)
                {
                    if (lowest > temp)
                    {
                        ff[i] = j;
                        lowest = temp;
                    }
                }
            }
        }
        frag[i] = lowest; bf[ff[i]] = 1; lowest = 10000;
    }
    printf("\nFile No\tFile Size \tBlock No\tBlock Size\tFragment");
    for(i=1;i<=nf && ff[i]!=0;i++)
        printf("\n%d\t%d\t%d\t%d\t%d", i, f[i], ff[i], b[ff[i]], frag[i]);
    getch();
}

```

- Dữ liệu đầu vào:

```
Enter the number of blocks : 3
Enter the number of files : 2
Enter the size of the blocks : -
Block 1 : 5
Block 2 : 2
Block 3 : 7
Enter the size of the files : -
File 1 : 1
File 2 : 4
```

- Kết quả chạy chương trình:

File No	File Size	Block No	Block	Size	Fragment
1	1	2	2	1	
2	4	1	5	1	

1.3. Phù hợp đầu tiên (First-fit)

- Code chương trình

```
#include<stdio.h>
#include<conio.h>
#define max 25
void main()
{
    int
        frag[max], b[max], f[max], i, j, nb, nf, temp, highes
        t = 0; static int bf[max], ff[max];
    clrscr();
    printf("\n\tMemory Management Scheme - Worst Fit");
    printf("\nEnter the number of blocks:");
    scanf("%d", &nb);
    printf("Enter the number of files:");
    scanf("%d", &nf);
    printf("\nEnter the size of the blocks:-\n");
    for (i = 1; i <= nb; i++)
    {
        printf("Block %d:", i);
        scanf("%d", &b[i]);
    }
    printf("Enter the size of the files :-\n");
    for (i = 1; i <= nf; i++)
    {
        printf("File %d:", i);
        scanf("%d", &f[i]);
    }
}
```

```

Page 28
for (i = 1; i <= nf; i++)
{
    for (j = 1; j <= nb; j++)
    {
        if (bf[j] != 1) //if bf[j] is not allocated
        {
            temp = b[j] - f[i];
            if (temp >= 0)
                if (highest < temp)
                {
                    frag[i] = highest; bf[ff[i]] = 1; highest = 0;
                }
            ff[i] = j; highest = temp;
        }
    }
    printf("\nFile_no:\tFile_size:\tBlock_no:\tBlock_size:\tFragement");
    for (i = 1; i <= nf; i++)
        printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d", i, f[i], ff[i], b[ff[i]], frag[i]);
    getch();
}

```

- Dữ liệu đầu vào:

```

Enter the number of blocks : 3
Enter the number of files : 2
Enter the size of the blocks : -
Block 1 : 5
Block 2 : 2
Block 3 : 7
Enter the size of the files : -
File 1 : 1
File 2 : 4

```

- Kết quả chạy chương trình:

File No	File Size	Block No	Block	Size	Fragment
1	1	3	7	6	
2	4	1	5	1	

2. Bài tập mẫu 2: Quản lý bộ nhớ

- Sử dụng con trỏ trong phân bổ bộ nhớ với hàm **malloc()**
 - o Cú pháp: **ptr = (castType*) malloc(size);**
 - o Con trỏ giữ địa chỉ byte đầu tiên trong bộ nhớ được phân bổ
- Phân bổ bộ nhớ liền kề với hàm **calloc()**
 - o Cú pháp: **ptr = (castType*)calloc(n, size);**
 - o Cấp phát và khởi tạo tất cả các bit thành zero với hai hàm **malloc ()** **calloc()**
- Giải phóng bộ nhớ với hàm **free()**
 - o Cú pháp: **free(ptr);**
 - o Câu lệnh giải phóng không gian được cấp phát trong bộ nhớ bởi con trỏ
- Thay đổi kích thước của bộ nhớ được phân bổ sử dụng dùng hàm **realloc()**
 - o Cú pháp: **ptr = realloc(ptr, x);**
 - o Câu lệnh phân bổ lại với một kích thước mới.

Yêu cầu: Viết chương trình trên HĐH Linux (ubuntu 16.04 LTS) minh họa bằng ngôn ngữ C việc phân bổ, giải phóng, quản lý bộ nhớ?

- Sử dụng con trỏ trong phân bổ và giải phóng bộ nhớ với hàm **malloc()**, **free()**.
- Chương trình mẫu: **malloc()**, **free()**.
- Code chương trình

```
#include <stdio.h>
#include <stdlib.h>

int main() {
    int n, i, *ptr, sum = 0;

    printf("Enter number of elements: ");
    scanf("%d", &n);

    ptr = (int*) malloc(n * sizeof(int));

    // if memory cannot be allocated
    if(ptr == NULL) {
        printf("Error! memory not allocated.");
        exit(0);
    }

    printf("Enter elements: ");
    for(i = 0; i < n; ++i) {
        scanf("%d", ptr + i);
        sum += *(ptr + i);
    }

    printf("Sum = %d", sum);

    // deallocating the memory
    free(ptr);

    return 0;
}
```

- Kết quả chạy chương trình:

```
Enter number of elements: 3
Enter elements: 100
20
36
Sum = 156
```

- Chương trình mẫu: **calloc()**, **free()**.

```
#include <stdio.h>
#include <stdlib.h>

int main() {
    int n, i, *ptr, sum = 0;
    printf("Enter number of elements: ");
    scanf("%d", &n);

    ptr = (int*) calloc(n, sizeof(int));
    if(ptr == NULL) {
        printf("Error! memory not allocated.");
        exit(0);
    }

    printf("Enter elements: ");
    for(i = 0; i < n; ++i) {
        scanf("%d", ptr + i);
        sum += *(ptr + i);
    }

    printf("Sum = %d", sum);
    free(ptr);
    return 0;
}
```

- Kết quả chạy chương trình:

```
Enter number of elements: 3
Enter elements: 100
20
36
Sum = 156
```


- Thay đổi kích thước của bộ nhớ được phân bổ sử dụng dùng hàm **realloc()**
- Chương trình mẫu: **realloc ()**.

```
#include <stdio.h>
#include <stdlib.h>

int main() {
    int *ptr, i , n1, n2;
    printf("Enter size: ");
    scanf("%d", &n1);

    ptr = (int*) malloc(n1 * sizeof(int));

    printf("Addresses of previously allocated memory:\n");
    for(i = 0; i < n1; ++i)
        printf("%pc\n", ptr + i);

    printf("\nEnter the new size: ");
    scanf("%d", &n2);

    // relocating the memory
    ptr = realloc(ptr, n2 * sizeof(int));

    printf("Addresses of newly allocated memory:\n");
    for(i = 0; i < n2; ++i)
        printf("%pc\n", ptr + i);

    free(ptr);

    return 0;
}
```

- Kết quả chạy chương trình:

```
Enter size: 2
Addresses of previously allocated memory:
26855472
26855476

Enter the new size: 4
Addresses of newly allocated memory:
26855472
26855476
26855480
26855484
```

3. Bộ nhớ chia sẻ (Share memory)

- Memory page structure: Struct page, Permission, Flag
- Library support: shm API, mmap, mmap API
- Sequence diagram to share memory via mmap API: create file fd, set memory size, map shared mem to process, unmap shared mem, remove the shared memory
- Example code

Yêu cầu: Viết chương trình trên HĐH Linux (ubuntu 16.04 LTS) minh họa bằng ngôn ngữ C việc phân bổ, giải phóng, quản lý bộ nhớ?

3.1. Bài tập mẫu 1

- **Yêu cầu:** Viết chương trình trên HĐH Linux (ubuntu 16.04 LTS) minh họa bằng ngôn ngữ C việc bộ nhớ chia sẻ (**Share memory**)?
- Chương trình mẫu:

```
#include <stdio.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <fcntl.h>
#include <sys/shm.h>
#include <sys/stat.h>
#include <sys/mman.h>
#include <sys/types.h>
#include <fcntl.h>
#include <unistd.h>

void main()
{
    int shm_fd;
    char read_buff[1024];
    void *ptr = NULL;

    shm_fd = shm_open("hello_ca_lop", O_RDONLY, 0666);
    ptr = mmap(0, 4096, PROT_READ, MAP_SHARED, shm_fd, 0);
    printf("%s", (char*)ptr);
    shm_unlink("hello_ca_lop");
}
```

- Kết quả chạy chương trình:

```

udoo@udoo-VirtualBox:~/fsoft$ cd ../
udoo@udoo-VirtualBox:~$ cd lop_thang_8/bai
bai1/ bai2/ bai3/ bai5/ bai6/ bai7/ bai8/ bai9/
udoo@udoo-VirtualBox:~$ cd lop_thang_8/bai^C
udoo@udoo-VirtualBox:~$ cd fsoft/bail0/mmap_sample/
udoo@udoo-VirtualBox:~/fsoft/bail0/mmap_sample$ ls
writer writer.c
udoo@udoo-VirtualBox:~/fsoft/bail0/mmap_sample$ vi reader.c
udoo@udoo-VirtualBox:~/fsoft/bail0/mmap_sample$ vi reader.c^C
udoo@udoo-VirtualBox:~/fsoft/bail0/mmap_sample$ gcc -o reader reader.c -lrt
udoo@udoo-VirtualBox:~/fsoft/bail0/mmap_sample$
udoo@udoo-VirtualBox:~/fsoft/bail0/mmap_sample$ ./reader
Hello World
udoo@udoo-VirtualBox:~/fsoft/bail0/mmap_sample$ █

```


4. Bài tập luyện tập

- Bài tập 1. Viết chương trình C minh họa việc khởi tạo một mảng kích thước 1MB?
- Bài tập 2. Viết chương trình C minh họa việc kiểm tra kích thước của một file?
- Bài tập 3. Viết chương trình C minh họa việc kiểm tra xem file có tồn tại không?

5. Bài thực hành ở nhà

Hãy thực hiện lại các bài tập sau bằng Windows Form hoặc C. Yêu cầu: Nộp lại source code vào thư mục trên kênh học tập Google Classroom.

- Bài tập 1. Viết chương trình minh họa việc tìm kiếm một tập tin?
- Bài tập 2. Viết chương trình minh họa việc kiểm tra tập tin có được tạo chưa?
- Bài tập 3. Viết chương trình minh họa việc kiểm tra việc thống kê bộ nhớ vật lý?
- Bài tập 3. Viết chương trình minh họa việc kiểm tra việc sử dụng bộ nhớ trong linux bằng GUI ?

Trường: ĐH CNTP TP.HCM Khoa: Công nghệ Thông tin Bộ môn: Mạng máy tính – Truyền thông MH: TH Hệ điều hành MSMH:	BÀI 9. QUẢN LÝ BỘ NHỚ ẢO	
--	---------------------------------	---

V. MỤC TIÊU

Sau khi hoàn thành bài thực hành này, sinh viên sẽ đạt được các kỹ năng sau:

- Viết được chương trình minh họa bằng ngôn ngữ C trong việc tạo, quản lý bộ nhớ giữa ảo:
- Triển khai kỹ thuật thay thế trang FIFO trên bộ nhớ ảo:
 - o FIFO, LRU, OPTIMAL(tối ưu)
 - o Đây là thuật toán quan trọng của quản lý bộ nhớ ảo
 - o Giúp HĐH quyết định trang bộ nhớ được di chuyển ra ngoài để tạo không gian cho trang cần thiết.
 - o Mục tiêu chính là giảm số lượng lỗi trang.
 - o LRU trong thuật toán này sẽ được thay thế , trang ít được sử dụng gần nhất
 - o OPTIMAL – các trang được thay thế sẽ không được sử dụng trong thời gian dài nhất. Ít lỗi trang so với các thuật toán trên
- Quản lý bộ nhớ với kỹ thuật phân vùng (MFT), triển khai và mô phỏng thuật toán MFT.
- Bộ nhớ được chia làm hai phần:
 - o Trong MFT, bộ nhớ được phân vùng thành các phân vùng kích thước cố định, mỗi công việc được gán cho một phân vùng
 - o Trong MVT, mỗi công việc chỉ nhận được lượng bộ nhớ mà nó cần. Sự phân vùng của bộ nhớ là động và thay đổi khi công việc vào và ra khỏi hệ thống.
 - o MFT gặp phải vấn đề phân mảnh bên trong và MVT gặp phải vấn đề phân mảnh bên ngoài

W. DỤNG CỤ - THIẾT BỊ THỰC HÀNH CHO MỘT SINH VIÊN:

STT	Chủng loại – quy cách vật tư	Số lượng	Đơn vị	Ghi chú
1	Computer	1	Bộ	

X. NỘI DUNG THỰC HÀNH

15. Tóm tắt lý thuyết

- Bài thực hành liên quan đến các vấn đề lý thuyết về khai báo, cấp phát, phân vùng và đọc bộ nhớ ảo (virtual memory) cụ thể như sau:
- Triển khai kỹ thuật thay thế trang trên bộ nhớ ảo:
 - o FIFO (First In First Out): đến trước về sau
 - o LRU (Least Recently Used)
 - o OPTIMAL: tối ưu
- Quản lý bộ nhớ với kỹ thuật phân vùng (MFT), triển khai và mô phỏng thuật toán MFT.

16. Bài thực hành trên lớp

16.1. Bài tập mẫu 1

Yêu cầu: Viết chương trình C minh họa việc triển khai kỹ thuật trên bộ nhớ ảo (virtual memory)?

Triển khai kỹ thuật thay thế trang FIFO trên bộ nhớ ảo

- Các bước thực hiện FIFO (First In First Out):
 1. Bắt đầu quá trình
 2. Đọc số trang n
 3. Đọc số trang không
 4. Đọc số trang vào một mảng a [i]
 5. Khởi tạo lịch phát sóng [i] = 0 để kiểm tra lượt truy cập trang
 6. Thay thế trang bằng hàng đợi tròn, trong khi đặt lại tính khả dụng của trang kiểm tra trong khung Đặt lịch phát sóng [i] = 1 nếu trang bị lỗi Số lượng trang trong khung
 7. In kết quả.
 8. Dừng quá trình.

```

#include<stdio.h>
#include<conio.h> int fr[3];
void main() {
    void display();
    int i, j, page[12] = { 2,3,2,1,5,2,4,5,3,2,5,2 };
    int
        flag1 = 0, flag2 = 0, pf = 0, frsize = 3, top = 0;
    clrscr();
    for (i = 0; i < 3; i++) {
        fr[i] =
            -1;
    }
    for (j = 0; j < 12; j++) {
        flag1 = 0; flag2 = 0; for (i = 0; i < 12; i++) {
            if (fr[i] == page[j]) {
                flag1 = 1; flag2 = 1; break;
            }
        }
        if (flag1 == 0) {
            for (i = 0; i < frsize; i++) {
                if (fr[i] ==
                    -1)
                {
                    fr[i] = page[j]; flag2 = 1; break;
                }
            }
        }
        if (flag2 == 0) {
            fr[top] = page[j];
            top++;
            pf++;
            if (top >= frsize)
                top = 0;
        }
        display();
    }
}

Page 31
    printf("Number of page faults : %d ", pf + frsize);
    getch();
}

void display()
{
    int i; printf("\n");
    for (i = 0; i < 3; i++)
        printf("%d\t", fr[i]);
}

```

- Kết quả đầu ra:

```
2 -1 -1
2 3 -1
2 3 -1
2 3 1
5 3 1
5 2 1
5 2 4
5 2 4
3 2 4
3 2 4
3 5 4
3 5 2
Number of page faults: 9
```

Triển khai kỹ thuật thay thế trang LRU(Least Recently Used) trên bộ nhớ ảo:

- Các bước thực hiện kỹ thuật thay thế trang **LRU**:
 1. Bắt đầu tiến trình
 2. Khai báo kích thước
 3. Nhận số lượng trang sẽ được chèn
 4. Nhận giá trị
 5. Khai báo bộ đếm và ngăn xếp
 6. Chọn trang ít được sử dụng gần đây nhất theo giá trị bộ đếm
 7. Xếp chồng chúng theo lựa chọn.
 8. Hiển thị các giá trị
 9. Dừng tiến trình
- Code bài toán:

```
#include<stdio.h>
#include<conio.h>
int fr[3];
void main()
{
    void display();
    int p[12] = { 2,3,2,1,5,2,4,5,3,2,5,2 }, i, j, fs[3];
    int index, k, l, flag1 = 0, flag2 = 0, pf = 0, frsize = 3;
    clrscr();
    for (i = 0; i < 3; i++)
    {
        fr[i] = -1;
    }
    for (j = 0; j < 12; j++)
    {
```

```

    flag1 = 0, flag2 = 0;
    for (i = 0; i < 3; i++)
    {
        if (fr[i] == p[j])
        {
            flag1 = 1;
            flag2 = 1; break;
        }
    }
    if (flag1 == 0)
    {
        for (i = 0; i < 3; i++) {
            if (fr[i] ==
                -1)
            {
                fr[i] = p[j]; flag2 = 1;
                break;
            }
        }
    }
    if (flag2 == 0) {
        for (i = 0; i < 3; i++)
            fs[i] = 0;
        for (k = j
            - 1, l = 1; l <= frsize
            - 1; l++, k--)
        {
            for (i = 0; i < 3; i++) {
                if (fr[i] == p[k]) fs[i] = 1;
            }
        }
        for (i = 0; i < 3; i++) {
            if (fs[i] == 0)
                index = i;
        }
        fr[index] = p[j];
        pf++;
    }
    display();
}
printf("
    \n no of page faults : % d",pf+frsize);
getch();
}

void display() {
    int i; printf("
        \
        n");
    for (i = 0; i < 3; i++)
        printf("
            \
            t % d",fr[i]);
}

```


- Kết quả đầu ra:

```

2 3 - 1
2 3 - 1
2 3 1
2 5 1
2 5 1
2 5 4
2 5 4
2 5 4
3 5 4
3 5 2
3 5 2
3 5 2
No of page faults : 7

```

Triển khai kỹ thuật thay thế trang tối ưu (OPTIMAL) trên bộ nhớ ảo

- Các bước thực hiện thuật toán thay thế trang tối ưu:
 1. Bắt đầu chương trình
 2. Đọc số trang và số khung
 3. Đọc mỗi giá trị trang
 4. Tìm kiếm trang trong khung
 5. nếu không có sẵn Phân bổ khung miễn phí
 6. Nếu không có khung nào thì miễn phí Replace trang với trang ít được sử dụng
 7. Số trang in lỗi trang
 8. Dừng quá trình.
- Code bài toán

```

#include<stdio.h>
#include<conio.h>
int fr[3], n, m;
void
display();
void main()
{
    int i, j, page[20], fs[10];
    int
        max, found = 0, lg[3], index, k, l, flag1 = 0, flag2 = 0, pf = 0;
    float pr;
    clrscr();
    printf("Enter length of the reference string: ");
    scanf("%d", &n);
    printf("Enter the reference string: ");
    for (i = 0; i < n; i++)
        scanf("%d", &page[i]);
    printf("Enter no of frames: ");
    scanf("%d", &m);
    for (i = 0; i < m; i++)
        fr[i] = -1; pf = m;
    Page 36

```

```
for (j = 0; j < n; j++) {
    flag1 = 0; flag2 = 0;
    for (i = 0; i < m; i++) {
        if (fr[i] == page[j]) {
            flag1 = 1; flag2 = 1;
            break;
        }
    }
    if (flag1 == 0) {
        for (i = 0; i < m; i++) {
            if (fr[i] ==
                -1)
            {
                fr[i] = page[j]; flag2 = 1;
                break;
            }
        }
    }
}

if (flag2 == 0) {
    for (i = 0; i < m; i++)
        lg[i] = 0;
    for (i = 0; i < m; i++) {
        for (k = j + 1; k <= n; k++) {
            if (fr[i] == page[k]) {
                lg[i] = k
                    - j;
                break;
            }
        }
    }
    found = 0;
    for (i = 0; i < m; i++) {
        if (lg[i] == 0) {
            index = i;
            found = 1;
            Page 37
            break;
        }
    }
}
```

```

        if (found == 0)
        {
            max = lg[0]; index = 0;
            for (i = 0; i < m; i++)
            {
                if (max < lg[i])
                {
                    max = lg[i];
                    index = i;
                }
            }
            fr[index] = page[j];
            pf++;
        }
        display();
    }
    printf("Number of page faults : %d\n", pf);
    pr = (float)pf / n * 100;
    printf("Page fault rate = %f \n", pr); getch();
}

void display()
{
    int i; for (i = 0; i < m; i++)
        printf("%d\t", fr[i]);
    printf("\n");
}

```

- Kết quả bài toán:

```

Enter length of the reference string : 12
Enter the reference string : 1 2 3 4 1 2 5 1 2 3 4 5
Enter no of frames : 3
1 - 1 - 1
1 2 - 1
1 2 3
1 2 4
1 2 4
1 2 4
1 2 5
1 2 5
1 2 5
3 2 5
4 2 5
4 2 5
Number of page faults : 7 Page fault rate = 58.333332

```

16.2. Bài tập mẫu 2

Yêu cầu: Viết chương trình C minh họa việc khai báo, cấp phát, phân vùng và đọc bộ nhớ ảo (virtual memory)?

Quản lý bộ nhớ với kỹ thuật phân vùng (MFT)

- Các bước thực hiện:

- + Bước 1: Bắt đầu quá trình.
- + Bước 2: Khai báo các biến.
- + Bước 3: Nhập tổng kích thước bộ nhớ ms.
- + Bước 4: Cấp phát bộ nhớ cho hệ điều hành. $Ms = ms - os$
- + Bước 5: Đọc phân vùng không được chia n, kích thước phân vùng = ms / n .
- + Bước 6: Đọc không tiến trình và kích thước tiến trình.
- + Bước 7: Nếu kích thước tiến trình nhỏ hơn kích thước phân vùng, thì phân bổ toàn bộ tiến trình. Trong khi cấp phát bộ nhớ cập nhật vùng trống và phân mảnh bên ngoài.

```
if (pn[i] == pn[j]) f = 1;
if (f == 0) {
    if (ps[i] <= siz)
    {
        extft = extft + size_1ps[i]; avail[i] = 1; count++;
    }
}
```

- + Bước 8: In kết quả

- Code bài toán:

```
1  #include<stdio.h>
2  #include<conio.h>
3  main()
4  {
5      int ms, bs, nob, ef, n,
6          mp[10], tif = 0; int i, p = 0;
7      //clrscr();
8      printf("Enter the total memory available (in Bytes) -- ");
9      scanf("%d", &ms);
10     printf("Enter the block size (in Bytes) -- ");
11     scanf("%d", &bs);
12     nob = ms / bs;
13     ef = ms - nob * bs;
14     printf("\nEnter the number of processes -- ");
15     scanf("%d", &n);
16     for (i = 0; i < n; i++)
17     {
18         printf("Enter memory required for process %d (in Bytes)-- ", i + 1);
19         scanf("%d", &mp[i]);
20     }
```

```

21     printf("\nNo. of Blocks available in memory--%d", nob);
22     printf("\n\nPROCESS\tMEMORYREQUIRED\tALLOCATED\tINTERNAL FRAGMENTATION");
23     for (i = 0; i < n && p < nob; i++)
24     {
25         printf("\n %d\t\t%d", i + 1, mp[i]);
26         if (mp[i] > bs)
27             printf("\t\tNO\t\t---");
28         else
29         {
30             printf("\t\tYES\t\t%d", bs - mp[i]);
31             tif = tif + bs - mp[i];
32             p++;
33         }
34     }
35     if (i < n)
36         printf("\nMemory is Full, Remaining Processes cannot be accomodated");
37     printf("\n\nTotal Internal Fragmentation is %d", tif);
38     printf("\nTotal External Fragmentation is %d", ef);
39     getch();
40 }
41

```

- Dữ liệu đầu vào:

```

Enter the total memory available(in Bytes)-- 1000
Enter the block size(in Bytes)-- 300
Enter the number of processes - 5
Enter memory required for process 1 (in Bytes)-- 275
Enter memory required for process 2 (in Bytes)-- 400
Enter memory required for process 3 (in Bytes)-- 290
Enter memory required for process 4 (in Bytes)-- 293
Enter memory required for process 5 (in Bytes)-- 100
No.of Blocks available in memory-- 3

```

- Kết quả chạy chương trình:

```

PROCESS ALLOCAT INTERNAL
MEMORY REQUIRED ED FRAGMENTATION
1 275 YES 25
2 400 NO---- -
3 290 YES 10
4 293 YES 7
Memory is Full, Remaining Processes cannot be accommodated Total
Internal Fragmentation is 42
Total External Fragmentation is 100

```

Viết chương trình mô phỏng thuật toán MVT.

- Các bước thực hiện:

- + Bước 1: Bắt đầu quá trình.
- + Bước 2: Khai báo các biến.
- + Bước 3: Nhập tổng kích thước bộ nhớ ms.
- + Bước 4: Cấp phát bộ nhớ cho hệ điều hành. $Ms = ms - os$
- + Bước 5: Đọc phân vùng không được chia n, kích thước phân vùng = ms / n .
- + Bước 6: Đọc không tiến trình và kích thước tiến trình.
- + Bước 7: Nếu kích thước tiến trình nhỏ hơn kích thước phân vùng, thì phân bổ toàn bộ tiến trình. Trong khi cấp phát bộ nhớ cập nhật vùng trống và phân mảnh bên ngoài.

```

if (pn[i] == pn[j]) f = 1;
if (f == 0) {
    if (ps[i] <= siz)
    {
        extft = extft + size_1ps[i]; avail[i] = 1; count++;
    }
}

```

- + Bước 8: In kết quả
- + Bước 9: Dừng quá trình.

- Yêu cầu:

```

#include<stdio.h>
#include<conio.h>
main()
{
    int ms, mp[10], i,
        temp, n = 0; char ch = 'y';
    clrscr();
    printf("\nEnter the total memory available (in Bytes)-- ");
    scanf("%d", &ms);
    temp = ms;
    for (i = 0; ch == 'y'; i++, n++)
    {
        printf("\nEnter memory required for process %d (in Bytes) -- ", i + 1);
        scanf("%d", &mp[i]);
        if (mp[i] <= temp)
        {
            printf("\nMemory is allocated for Process %d ", i + 1);
            temp = temp - mp[i];
        }
        else
        {
            printf("\nMemory is Full"); break;
        }
        printf("\nDo you want to continue(y/n) -- ");
        scanf(" %c", &ch);
    }
}

```

```

    }
    printf("\n\nTotal Memory Available -- %d", ms);
    printf("\n\n\tPROCESS\t\t MEMORY ALLOCATED ");
    for (i = 0; i < n; i++)
        printf("\n \t%d\t\t\t%d", i + 1, mp[i]);
    printf("\n\nTotal Memory Allocated is %d", ms - temp);
    printf("\nTotal External Fragmentation is %d", temp);
    getch();
}

```

- Dữ liệu đầu vào:

```

Enter the total memory available(in Bytes) - 1000
Enter memory required for process 1 (in Bytes) - 400
Memory is allocated for Process 1
Do you want to continue(y / n) --y
Enter memory required for process 2 (in Bytes)-- 275
Memory is allocated for Process 2
Do you want to continue(y / n) - y
Enter memory required for process 3 (in Bytes) - 550

```

- Kết quả chạy chương trình:

```

Memory is Full
Total Memory Available - 1000
PROCESS MEMORY ALLOCATED
1 400
2 275
Total Memory Allocated is 675
Total External Fragmentation is 325

```

17. Bài tập luyện tập

Bài tập 1. Viết chương trình C mô phỏng các kỹ thuật cấp phát bộ nhớ liên kế cho một tiến trình hay một không gian lưu trữ?

Bài tập 2. Viết chương trình C minh họa việc bộ nhớ bất kỳ di chuyển ra ngoài để tạo không gian cho trang hiện tại đang cần thiết?

Bài tập 3. Viết chương trình C minh họa việc trang phía trước hàng đợi được chọn để xóa ?

Bài tập 4. Viết chương trình C minh họa việc khởi tạo một mảng kích thước 1MB?

Bài tập 5. Viết chương trình C minh họa việc kiểm tra kích thước của một file?

Bài tập 6. Viết chương trình C minh họa việc kiểm tra xem file có tồn tại không?

18. Bài thực hành ở nhà

Hãy thực hiện lại các bài tập sau bằng Windows Form hoặc C.

Yêu cầu: Nộp lại source code vào thư mục trên kênh học tập Google Classroom.

Bài tập 1. Viết chương trình minh họa việc tìm kiếm một tập tin?

Bài tập 2. Viết chương trình minh họa việc kiểm tra tập tin có được tạo chưa?

Bài tập 3. Viết chương trình minh họa việc kiểm tra việc thống kê bộ nhớ vật lý?

Bài tập 3. Viết chương trình minh họa việc kiểm tra việc sử dụng bộ nhớ trong linux bằng GUI ?

Bài tập 4: Hoàn chỉnh chương trình C mô phỏng các kỹ thuật cấp phát bộ nhớ liên kê ?



Y. MỤC TIÊU

Sau khi hoàn thành bài thực hành này, sinh viên sẽ đạt được các kỹ năng sau:

- Viết được chương trình minh họa tương tác với tập tin bằng ngôn ngữ C.
- Áp dụng được các ngôn ngữ lập trình khác nhau để viết minh họa hàm.
- Hình thành tư duy xử lý bài toán lập trình.

Z. DỤNG CỤ - THIẾT BỊ THỰC HÀNH CHO MỘT SINH VIÊN:

STT	Chủng loại – quy cách vật tư	Số lượng	Đơn vị	Ghi chú
1	Computer	1	Bộ	

AA. NỘI DUNG THỰC HÀNH

19. Tóm tắt lý thuyết

Bài thực hành liên quan đến các vấn đề lý thuyết về tập tin, cụ thể như sau:

- Kỹ thuật tổ chức tập tin một cấp
- Kỹ thuật tổ chức tập tin 2 cấp
- Kỹ thuật cấp phát tập tin tuần tự
- Kỹ thuật cấp phát tập tin theo chuỗi liên kết

20. Bài thực hành trên lớp

20.1. Bài tập mẫu 1

Yêu cầu: Viết chương trình C mô phỏng kỹ thuật tổ chức tập tin một cấp.

Mô tả vấn đề: Cấu trúc thư mục là kỹ thuật tổ chức tập tin một cấp. Trong kỹ thuật tổ chức tập tin một cấp tất cả các tệp được đặt trong một thư mục. Có một thư mục gốc có tất cả các tập tin. Nó có một kiến trúc đơn giản và không có thư mục con. Lợi thế của cấp độ đơn hệ thống thư mục là nó rất dễ dàng để tìm thấy một tập tin trong thư mục.

Chương trình mẫu:

```
#include <stdio.h>
struct
{
    char dname[10], fname[10][10];
    int fcnt;
}dir;
void main()
{
    int i,ch;
    char f[30];
    clrscr();
    dir.fcnt = 0;
    printf("\nEnter name of directory -- ");
    scanf("%s", dir.dname);
    while(1)
    {
        printf("\n\n1. Create File \t2. Delete File \t3. Search File \n 4.
Display Files \t5. Exit \nEnter your choice -- ");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1: printf("\nEnter the name of the file -- ");
                    scanf("%s",dir.fname[dir.fcnt]);
                    dir.fcnt++;
                    break;
            case 2: printf("\nEnter the name of the file -- ");
                    scanf("%s",f);
                    for (i=0; i<dir.fcnt; i++)
                    {
                        if(strcmp(f, dir.fname[i])==0)
                        {
                            printf("File %s is deleted ",f);
                            strcpy(dir.fname[i],dir.fname[dir.fcnt-1]);
                            break;
                        }
                    }
                    if(i==dir.fcnt)
                        printf("File %s not found",f);
                    else dir.fcnt--;
                    break;
            case 3: printf("\nEnter the name of the file -- ");
                    scanf("%s",f);
                    for (i=0; i<dir.fcnt; i++)
                    {
                        if(strcmp(f, dir.fname[i])==0)
                        {
                            printf("File %s is found ",f);
                            break;
                        }
                    }
                    if(i==dir.fcnt)
                        printf("File %s not found",f);
                    break;
            case 4: if(dir.fcnt==0)
                    printf("\nDirectory Empty");
                    else
                    {
                        printf("\nThe Files are -- ");
                        for(i=0;i<dir.fcnt; i++)
                            printf( "\t %s", dir.fname[i]);
                    }
                    Break;
            Default: exit(0);
        }
    }
    getch();}
```

Kết quả chạy chương trình:

Enter name of directory -- CSE

1. Create File 2. Delete File 3. Search File

4. Display Files 5. Exit Enter your choice – 1

Enter the name of the file -- A

1. Create File 2. Delete File 3. Search File

4. Display Files 5. Exit Enter your choice – 1

Enter the name of the file -- B

1. Create File 2. Delete File 3. Search File
 4. Display Files 5. Exit Enter your choice – 1
 Enter the name of the file -- C

1. Create File 2. Delete File 3. Search File
 4. Display Files 5. Exit Enter your choice – 4
 The Files are -- A B C

1. Create File 2. Delete File 3. Search File
 4. Display Files 5. Exit Enter your choice – 3
 Enter the name of the file – ABC
 File ABC not found

1. Create File 2. Delete File 3. Search File
 4. Display Files 5. Exit Enter your choice – 2
 Enter the name of the file – B
 File B is deleted

1. Create File 2. Delete File 3. Search File
 4. Display Files 5. Exit Enter your choice – 5

20.2. Bài tập mẫu 2

Yêu cầu: Viết chương trình C mô phỏng kỹ thuật tổ chức tập tin hai cấp

Mô tả vấn đề: Trong hệ thống thư mục hai cấp, mỗi người dùng có thư mục tập người dùng (UFD) riêng. Hệ thống duy trì một khối chính có một mục nhập cho mỗi người dùng. Khối chính này chứa địa chỉ của thư mục của người dùng. Khi người dùng bắt đầu công việc của hoặc người dùng đăng nhập, hệ thống của thư mục tập chính (MFD) được tìm kiếm. Khi người dùng tham chiếu đến một tệp cụ thể, chỉ UFD của riêng họ được tìm kiếm

Chương trình mẫu:

```

#include <stdio.h>
struct
{
    char dname[10], fname[10][10];
    int fcnt;
} dir[10];
void main()
{
    int i, ch, dcnt, k;
    char f[30], d[30];
    clrscr();
    dcnt = 0;
    while(1)
    {
        printf("\n\n1. Create Directory\t2. Create File\t3. Delete
File ");
        printf("\n 4. Search Files\t\t5. Display\t6.Exit\t Enter
your choice -- ");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1: printf("\nEnter the name of the file -- ");
                    scanf("%s",dir[dcnt].dname);
                    dir[dcnt].fcnt=0;
                    dcnt++;
                    printf("Directory created");
                    break;
            case 2: printf("\nEnter name of the directory -- ");
                    scanf("%s",d);
                    for (i=0 ; i<dcnt; i++)
                        if(strcmp(d,dir[i].dname)==0)
                        {
                            printf("Enter name of the file -- ");
                            scanf("%s",dir[i].fname[dir[i].fcnt]);
                            dir[i].fcnt++;
                        }

```

```

        printf("File created");
    }
    if(i==dcnt)
        printf("Directory %s not found",d);
    break;

case 3: printf("\nEnter name of the directory -- ");
    scanf("%s",d);
    for (i=0; i<dcnt; i++)
    {
        if(strcmp(d,dir[i].dname)==0)
        {
            printf("Enter name of the file -- ");
            scanf("%s",f);
            for (k=0; k<dir[i].fcnt; k++)
            {
                if(strcmp(f, dir[i].fname[k])==0)
                {
                    printf("File %s is deleted ",f);
                    dir[i].fcnt--;

                    strcpy(dir[i].fname[k],dir[i].fname[dir[i].fcnt]);
                    goto jmp;
                }
            }
            printf("File %s not found",f); goto jmp;
            break;
        }
    }
    printf("Directory %s not found",d);
    jmp : break;

case 4: printf("\nEnter name of the directory -- ");
    scanf("%s",d);
    for (i=0; i<dcnt; i++)
    {

```

```

        if(strcmp(d,dir[i].dname)==0)
        {
            printf("Enter name of the file -- ");
            scanf("%s",f);
            for (k=0; k<dir[i].fcnt; k++)
            {
                if(strcmp(f, dir[i].fname[k])==0)
                {
                    printf("File %s is found ",f);
                    goto jmp1;
                }
            }
            printf("File %s not found",f); goto jmp1;
        }
    }
    printf("Directory %s not found",d); jmp1: break;
case 5: if(dcnt==0)
    printf("\nNo Directory's ");
else
{
    printf("\nDirectory\tFiles");
    for (i=0; i<dcnt; i++)
    {
        printf("\n%s\t\t",dir[i].dname);
        for (k=0; k<dir[i].fcnt; k++)
            printf("\t%s",dir[i].fname[k]);
    }
    break;
Default: exit(0);
}
getch();
}

```

Kết quả chạy chương trình:

1.Create Directory 2. Create File 3. Delete File

4. Search File 5. Display 6. Exit

Enter your choice – 1

Enter name of directory -- DIR1 Directory created

1.Create Directory 2. Create File 3. Delete File

4. Search File 5. Display 6. Exit Enter your choice – 1

Enter name of directory -- DIR2 Directory created
 1.Create Directory 2. Create File 3. Delete File
 4. Search File 5. Display 6. Exit Enter your choice – 2
 Enter name of the directory – DIR1
 Enter name of the file -- A1
 File created
 1.Create Directory 2. Create File 3. Delete File
 4. Search File 5. Display 6. Exit
 Enter your choice – 2
 Enter name of the directory – DIR1
 Enter name of the file -- A2
 File created
 1.Create Directory 2. Create File 3. Delete File
 4. Search File 5. Display 6. Exit
 Enter your choice – 6

20.3. Bài tập luyện tập

Bài tập 1. Viết 1 chương trình C để cấp phát tập tin tuần tự

Mô tả vấn đề: cấp phát tập tin tuần tự là phương pháp phổ biến, được sử dụng cho các bản ghi. Tất cả các bản ghi (của hệ thống) có cùng độ dài, bao gồm cùng một số trường độ dài cố định theo một thứ tự cụ thể. Vì chiều dài và vị trí của từng trường được biết, chỉ giá trị của các trường cần được lưu trữ, tên trường và độ dài cho mỗi trường là thuộc tính của cấu trúc tệp

Thuật toán:

Bước 1: Khởi động chương trình

Bước 2: Nhập số lượng tập tin

Bước 3: Nhập dung lượng bộ nhớ cấp cho mỗi tập tin

Bước 4: Phân bổ theo tuần tự.

Chọn ngẫu nhiên 1 location với $s1 = \text{random}(100)$

a. Kiểm tra location có trống hông?

b. Tiến hành phân bổ và đặt cờ

Bước 5: In tập tin, chiều dài tập tin và Block đã phân bổ

Bước 6: Đóng chương trình

Bài tập 2. Viết 1 chương trình C để cấp phát tập tin bằng phương pháp chuỗi

Mô tả vấn đề: Trong bảng phân bổ tập tin, phương pháp chuỗi có một con trỏ đến Block bắt đầu. Bên trong mỗi Block có một con trỏ chỉ đến Block kế tiếp. Với cấp phát theo chuỗi sẽ không có sự phân mảnh bên ngoài.

Thuật toán:

Bước 1: Khởi động chương trình.

Bước 2: Lấy số lượng tệp.

Bước 3: Nhập yêu cầu bộ nhớ của từng tệp.

Bước 4: Phân bổ các location cần thiết bằng cách chọn một location ngẫu nhiên $q = \text{random}(100)$;

a) Kiểm tra xem location đã chọn có rảnh không.

b) Nếu location rảnh thì đặt cờ = 1 cho các location được cấp phát.

Bước 5: In tập tin, chiều dài tập tin và Block đã phân bổ

Bước 6: Dừng chương trình

Bài tập 3. Viết chương trình C mô tả việc cấp phát vùng nhớ, có kiểm tra trạng thái bộ nhớ đã cấp phát

21. Bài thực hành ở nhà

Hãy thực hiện bài tập sau bằng C#. Yêu cầu: Nộp lại source code vào thư mục trên kênh học tập Google Classroom.

Bài tập 1. **Viết 1 chương trình C để cấp phát tập tin tuần tự.**

Bài tập 2. **Viết 1 chương trình C để cấp phát tập tin bằng phương pháp chuỗi**

Bài tập 3. **Viết chương trình C mô tả việc cấp phát vùng nhớ, có kiểm tra trạng thái bộ nhớ đã cấp phát**