

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN



ĐỒ ÁN CUỐI KỲ
HỆ ĐIỀU HÀNH

Hệ điều hành 19_3 | 12/01/2022

Mục lục

I. Báo cáo nhóm	2
<i>I.1. Thông tin nhóm</i>	<i>2</i>
<i>I.2. Bảng phân công.....</i>	<i>2</i>
<i>I.3. Kết quả đồ án (Đã làm được và Chưa làm được).....</i>	<i>2</i>
II. Báo cáo đồ án	2
<i>II.1. Câu 1, kỹ thuật phân trang và bộ nhớ ảo kèm các thông số khảo sát.</i>	<i>2</i>
II.1.1. Kỹ thuật phân trang	2
II.1.2. Bộ nhớ ảo	6
II.1.3. Khảo sát các thông số về bộ nhớ trên máy tính cá nhân	8
<i>II.2. Câu 2</i>	<i>12</i>
II.2.1. Thông tin sơ lược về chương trình	12
II.2.2. Chương trình C	13
II.2.3. Chương trình P	16
III. Tham khảo.....	20

I. Báo cáo nhóm

I.1. Thông tin nhóm

Đề án cuối kỳ này được thực hiện bởi nhóm gồm 3 sinh viên, thông tin từng thành viên bao gồm tên và mã số sinh viên như sau:

1. Trần Vũ Việt Cường – 19120465
2. Phạm Thành Đạt – 19120473
3. Trần Bảo Tín – 19120684

I.2. Bảng phân công

Thành viên	Các công việc được phân công
Trần Vũ Việt Cường	Thực hiện câu 1 của đề án bao gồm việc “Trình bày kỹ thuật phân trang và bộ nhớ ảo”, khảo sát các thông số về bộ nhớ được yêu cầu trên máy tính cá nhân, tổng hợp báo cáo
Phạm Thành Đạt	Thực hiện chương trình P của câu 2, viết báo cáo, quay video
Trần Bảo Tín	Thực hiện chương trình C của câu 2, viết báo cáo, quay video

I.3. Kết quả đề án (Đã làm được và Chưa làm được)

Thành viên	Đã hoàn thành	Chưa làm được	Tự đánh giá
Trần Vũ Việt Cường	Trình bày kỹ thuật phân trang và bộ nhớ ảo	Không	100%
	Khảo sát đầy đủ các thông số được yêu cầu		
Phạm Thành Đạt	Các chức năng của chương trình P và xử lý critical section	Không	100%
Trần Bảo Tín	Các chức năng của chương trình C và xử lý tiến trình song song	Không	100%

II. Báo cáo đề án

II.1. Câu 1, kỹ thuật phân trang và bộ nhớ ảo kèm các thông số khảo sát.

II.1.1. Kỹ thuật phân trang

II.1.1.1. Định nghĩa

Phân trang là một kỹ thuật quản lý bộ nhớ mà trong đó, bộ nhớ vật lý được chia thành các khối nhỏ có kích thước cố định và bằng nhau được gọi là các *khung trang* (page frame). Không gian địa chỉ logic của các tiến trình cũng được chia

thành những khối có kích thước bằng với kích thước của khung, được gọi là các *trang* (page). Mỗi tiến trình khi chạy sẽ được cấp phát cho các khung trang để chứa các trang của tiến trình, và như vậy thì một tiến trình có thể nằm ở các vị trí khác nhau (không liền kề) ở trong bộ nhớ chứ không bắt buộc phải nằm liền kề nhau theo thứ tự của các trang. (chèn vd ở đây). Kích thước của trang và khung trang luôn là một lũy thừa của 2 (điều này giúp việc xác định và ánh xạ địa chỉ logic sang địa chỉ vật lý có thể thực hiện dễ dàng).

Kỹ thuật phân trang hạn chế được nhược điểm của kỹ thuật cấp phát liên tục (là hiện tượng phân mảnh bộ nhớ), phương pháp này không bị phân mảnh ngoài vì từng khung nhớ trong bộ nhớ đều có thể được sử dụng để cấp phát cho tiến trình.

II.1.1.2. Bảng phân trang

Do bản chất của kỹ thuật là tiến trình sẽ nằm trong các khung nhớ không liền kề nhau, để ánh xạ đại chỉ logic của tiến trình thành địa chỉ vật lý thì không chỉ cần dùng thanh ghi cơ sở, vì đó là không đủ. Do vậy, hệ điều hành sử dụng *bảng trang* (page table).

Các ô của bảng trang tương ứng với một trang và thông tin ở ô đó là địa chỉ cơ sở của khung chứa trang đó. Mỗi tiến trình sẽ có một bảng trang riêng. (chèn vd bảng trang).

II.1.1.3. Ánh xạ địa chỉ

II.1.1.3.a. Địa chỉ

Trong kỹ thuật phân trang, một địa chỉ logic bao gồm hai thành phần là *số thứ tự trang* (p) và *độ dịch hay địa chỉ tương đối* (o) của địa chỉ tính từ đầu trang đó. Số thứ tự trang được dùng để tìm ra ô tương ứng trong bảng trang, từ đó là tìm được địa chỉ cơ sở của khung trang tương ứng. Địa chỉ này được cộng vào độ dịch trong trang để tạo ra địa chỉ vật lý và được chuyển cho mạch điều khiển bộ nhớ để truy cập.

Ví dụ:

Địa chỉ logic
Độ dài:

Số thứ tự trang (p)	Độ dịch trong trang (o)
m	n

Ví dụ:

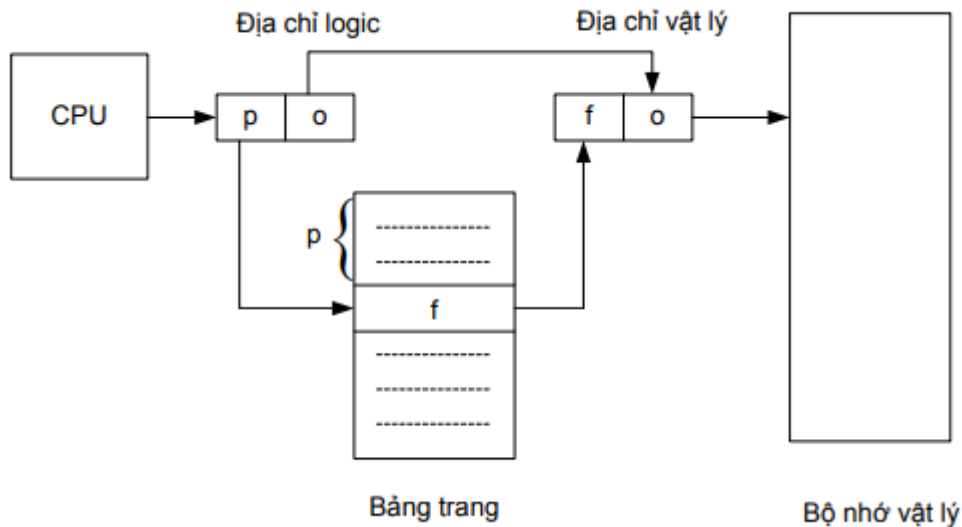
Một hệ thống máy tính có kích thước trang (pagesize) là 1024B, địa chỉ logic của độ dài là 16bit. Xác định p và o ở địa chỉ logic 1502,

- Dạng nhị phân của địa chỉ 1502 là: 0000010111011110
- $p = \lfloor 1502/1024 \rfloor = 1$, tương ứng với 6 bit đầu tiên: 000001
- $o = 1502 \bmod 1024 = 478$ tương ứng với 10 bit còn lại: 0111011110

Như vậy, địa chỉ logic 1502 ở dạng thập phân được chia làm 2 phần với 6 bit để chỉ thứ tự trang và 10 bit để chỉ độ dịch so với trang.

II.1.1.3.b. Cơ chế ánh xạ địa chỉ

Ánh xạ địa chỉ đều phải dựa vào phân cứng vì thao tác này cần thực hiện thường xuyên và ảnh hưởng lớn đến hiệu suất cũng như tốc độ của cả một hệ thống.



II.1.1.3.c. Quản lý khung

Với việc cấp phát một cách không liên tục các khung cho các trang nhớ, bắt buộc yêu cầu hệ điều hành phải có một cách để kiểm soát một khung đã được cấp phát hay chưa. Để quản lý được các thông tin này, hệ điều hành sử dụng một *bảng khung* (frame table). Bảng khung chứa các thông tin cơ bản về một khung như đã được cấp phát hay chưa, nếu đã cấp phát thì khung đó đã được cấp phát cho tiến trình nào, ...

II.1.1.4. Xây dựng bảng phân trang

Việc thao tác với bộ nhớ đều đòi hỏi thực hiện thao tác ánh xạ địa chỉ logic và địa chỉ vật lý. Do đó, tốc độ ảnh hưởng cực kỳ lớn đến hiệu suất của toàn bộ hệ thống. Việc tìm cách lưu trữ bảng phân trang một cách hợp lý để tăng tốc độ truy xuất cũng như thao tác nhanh nhất là cần thiết.

II.1.1.4.a. Lưu trữ page tables trong các thanh ghi

Cách này sử dụng một tập hợp các thanh ghi dành riêng cho việc lưu bảng trang. Các thanh ghi này được thiết kế riêng cho việc chứa dữ liệu của bảng trang cũng như phù hợp cho thao tác ánh xạ địa chỉ bộ nhớ. Việc truy cập các thanh ghi này được phân quyền sao cho chỉ có hệ điều hành chạy trong nhân mới có thể truy cập được. Ưu điểm của cách này là có thể đảm bảo được tốc độ rất cao, nhược điểm thì lại bị hạn chế số lượng bảng phân trang vì số lượng thanh ghi của một hệ thống máy tính bình thường thì không nhiều. Trước đây, trong các hệ thống máy tính sử dụng kỹ thuật lưu bảng trang trong các thanh ghi thì số lượng ô trong bảng trang thường không vượt quá 256.

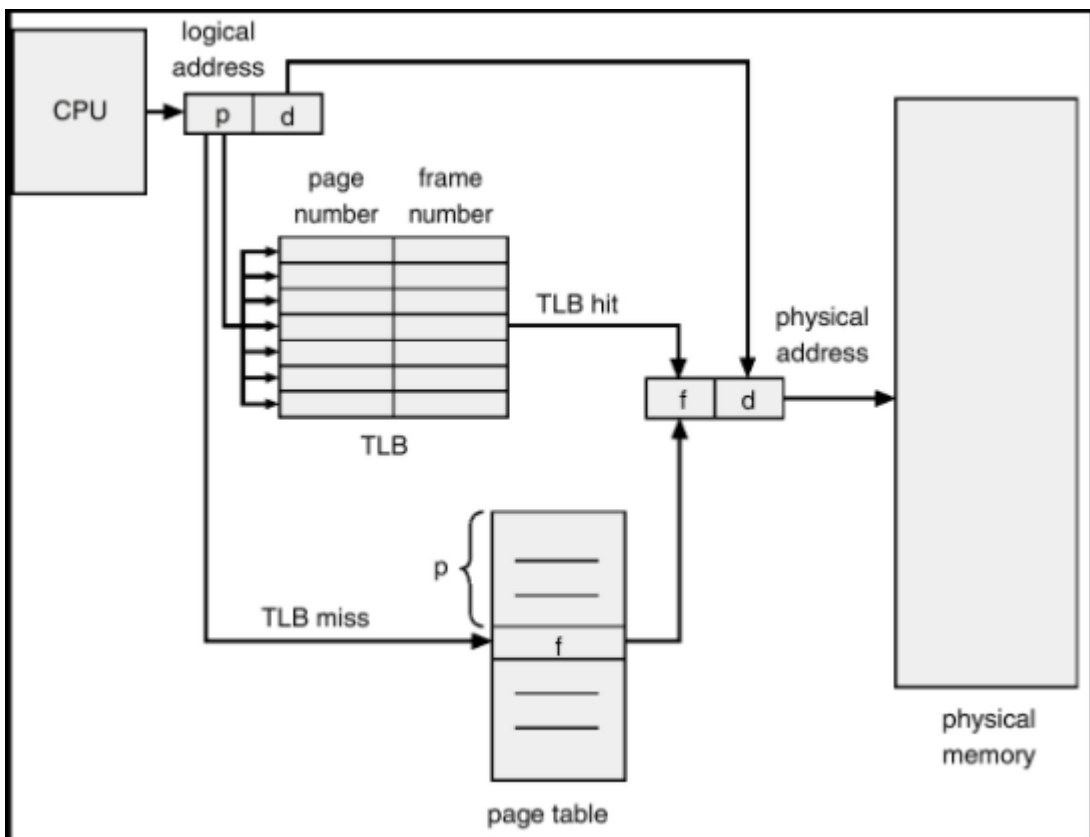
II.1.1.4.b. Lưu trữ page tables trong bộ nhớ trong

Các máy tính hiện đại ngày càng phát triển kèm theo bộ nhớ lớn, dẫn đến các bảng phân trang cũng phải lớn dần theo tương ứng. Vì vậy việc lưu trữ bảng trang trong các thanh ghi là một cách không còn phù hợp với yêu cầu sử dụng của máy tính hiện đại. Khi đó, một giải pháp khác được thay thế đó là lưu các bảng trang trong bộ nhớ.

Trong bộ nhớ, vị trí của mỗi bảng trang sẽ được trỏ tới bởi một thanh ghi được gọi là thanh ghi cơ sở của bảng trang (Page Table Base Register - PTBR). Như vậy, mỗi lần truy cập vào bộ nhớ trong đòi hỏi hai thao tác truy cập, một là đọc ô nhớ tương ứng trong bảng trang và hai là để thực hiện việc truy cập cần thiết. Do đó dẫn đến tốc độ bị ảnh hưởng đáng kể, cần nhiều thời gian cho việc truy cập bảng.

II.1.1.4.c. Bộ nhớ đệm cache

Đây là giải pháp nhằm giảm nhược điểm của việc lưu trữ page table trong bộ nhớ trong. Cách này sử dụng cache tốc độ cao chuyên dụng Translation Look-aside Buffer – TLB. Mỗi phần tử trong TLB được lưu trữ dưới dạng key – value. Key ở đây là số thứ tự trang và value số thứ tự khung trang tương ứng. Khi đưa vào TLB một số thứ tự trang thì nó sẽ được so sánh với toàn bộ khóa đang có trong TLB, nếu có giá trị trùng khớp thì giá trị là số thứ tự khung trang tương ứng sẽ được trả ra. Vì dữ liệu được lưu trữ dưới dạng key – value, việc tìm kiếm khóa và giá trị diễn ra cực kỳ nhanh.



TLB mang lại độ hiệu quả gần như bằng với bộ nhớ đệm thông thường. Các CPU

ngày nay đều sử dụng TLB nhiều mức để tăng hiệu quả sử dụng và hiệu suất của hệ thống.

II.1.1.4.d. Bảng trang nhiều mức

Hiện nay, các máy tính hiện đại đều cho phép sử dụng không gian địa chỉ logic rất lớn (các hệ thống 32-64 bit) dẫn đến kích thước các bảng trang cũng phải lớn dần theo. Ví dụ với hệ thống Windows 32bit, pagesize = 4KB thì sau một vài tính toán cụ thể ta sẽ tính được kích thước bảng phân trang sẽ rơi vào khoảng 4MB, mà mỗi tiến trình đều sẽ cần một bảng phân trang như vậy, hiển nhiên đây là một con số vô cùng lớn.

Một giải pháp được đưa ra là chia nhỏ bảng trang thành các thành phần nhỏ hơn sao cho các phần nhỏ có thể được lưu trong bộ nhớ một cách độc lập với nhau. Ưu điểm của phương pháp này là việc cấp phát bộ nhớ để lưu các phần nhỏ của bảng trang sẽ dễ dàng hơn việc cấp phát cho cả một bảng trang lớn, thêm nữa là việc chia nhỏ bảng trang có thể tận dụng tối đa bảng trang và giảm đi sự dư thừa (ở trường hợp kích thước của tiến trình nhỏ hơn nhiều so với kích thước của một bảng trang).

II.1.2. Bộ nhớ ảo

II.1.2.1. Khái niệm

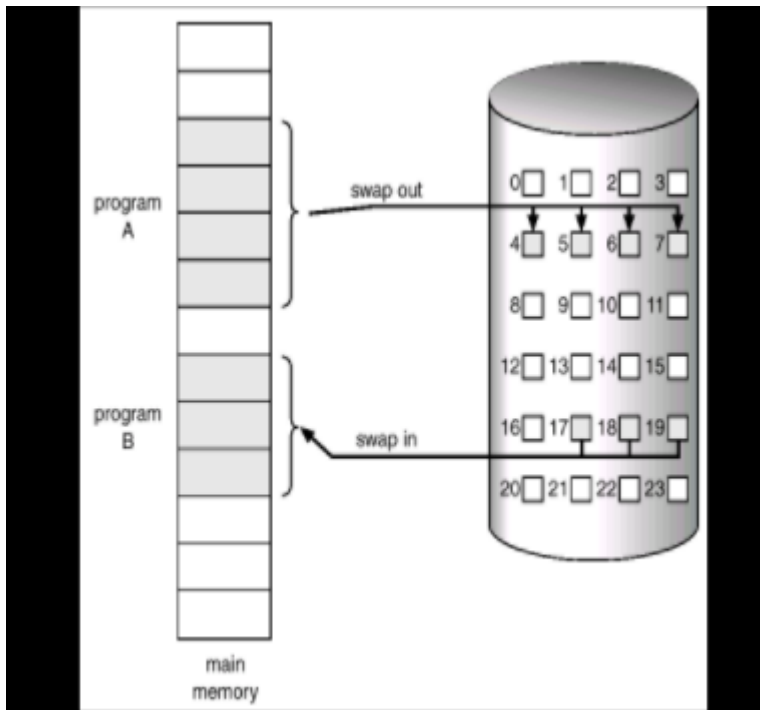
Bộ nhớ ảo là một phương pháp tổ chức quan trọng được sử dụng trong hầu hết các hệ điều hành hiện nay. Nó đơn giản hóa rất nhiều nhiệm vụ của coder và cho phép sử dụng một không gian bộ nhớ lớn hơn nhiều so với không gian nhớ thực sự. Bộ nhớ ảo là bộ nhớ logic theo một cách nhìn nào đó, tiến trình sẽ không bị hạn chế bởi bộ nhớ thực. Bộ nhớ ảo lớn hơn rất nhiều so với bộ nhớ thật, bao gồm cả không gian trên đĩa.

Bộ nhớ ảo phần lớn được xây dựng dựa trên hệ thống phân trang, các trang là đơn vị để nạp từ đĩa vào khi cần.

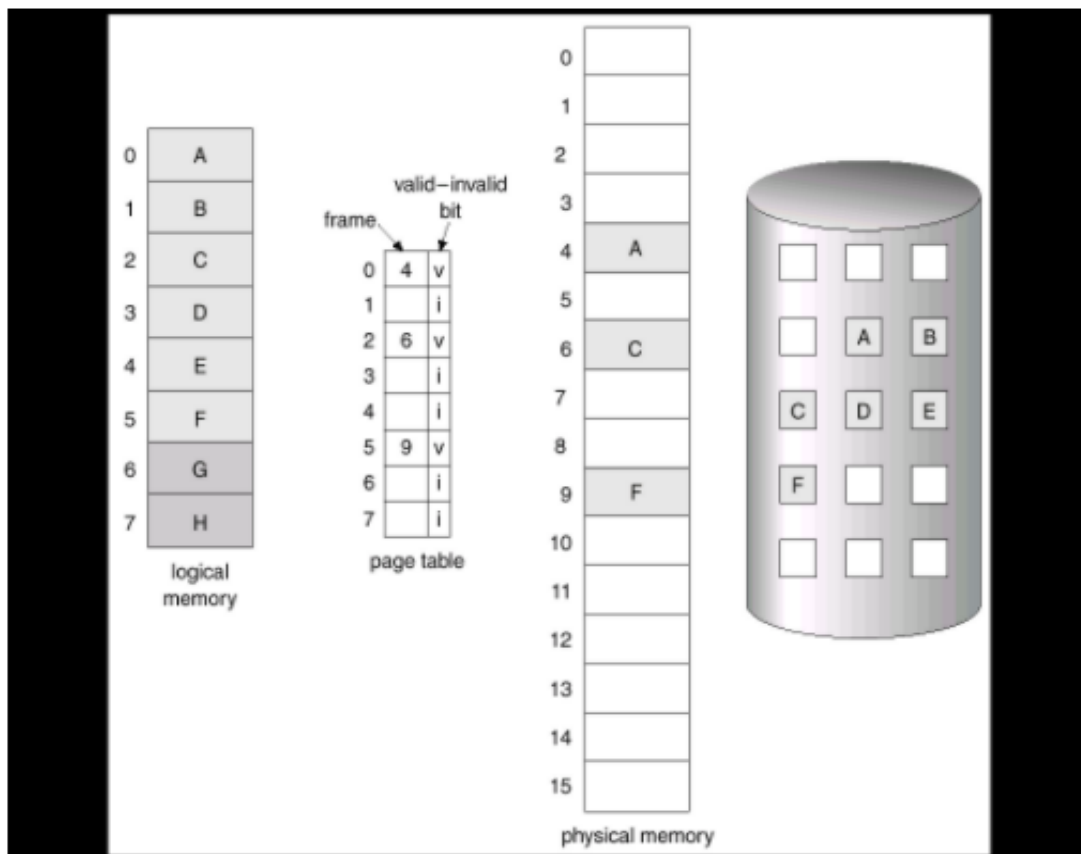
II.1.2.2. Demand Paging

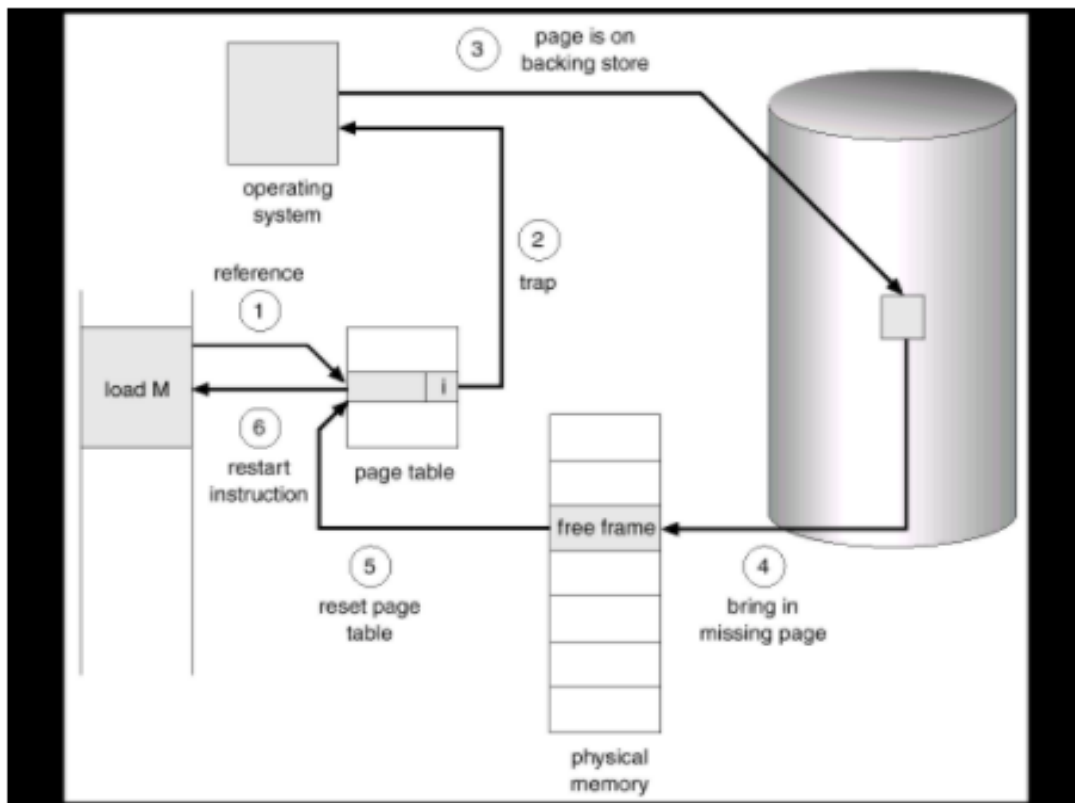
Là kỹ thuật phân trang kết hợp với trao đổi bộ nhớ - đĩa. Tiến trình sẽ được phân trang và lưu trên đĩa, khi cần thực hiện nó, nó sẽ được nạp vào trong bộ nhớ. Tuy nhiên, chỉ có các trang cần thiết của tiến trình mới được nạp vào, đây chính là nạp theo nhu cầu (demand paging)

Để xác định một trang đã được nạp vào bộ nhớ hay chưa, ta sử dụng thêm một bit P với giá trị 1(0) tương ứng với việc đã được nạp vào bộ nhớ hay chưa.



Khi tiến trình truy cập đến một trang đã ở trong bộ nhớ (nhận biết qua việc kiểm tra bit P), thì việc truy cập diễn ra bình thường (hình trên). Nhưng nếu trang đó chưa được nạp vào bộ nhớ, thì một sự kiện “lỗi trang” sẽ phát sinh, lệnh thực hiện tiến trình sẽ bị ngắt việc xử lý sự kiện này được mô tả ở hình dưới:



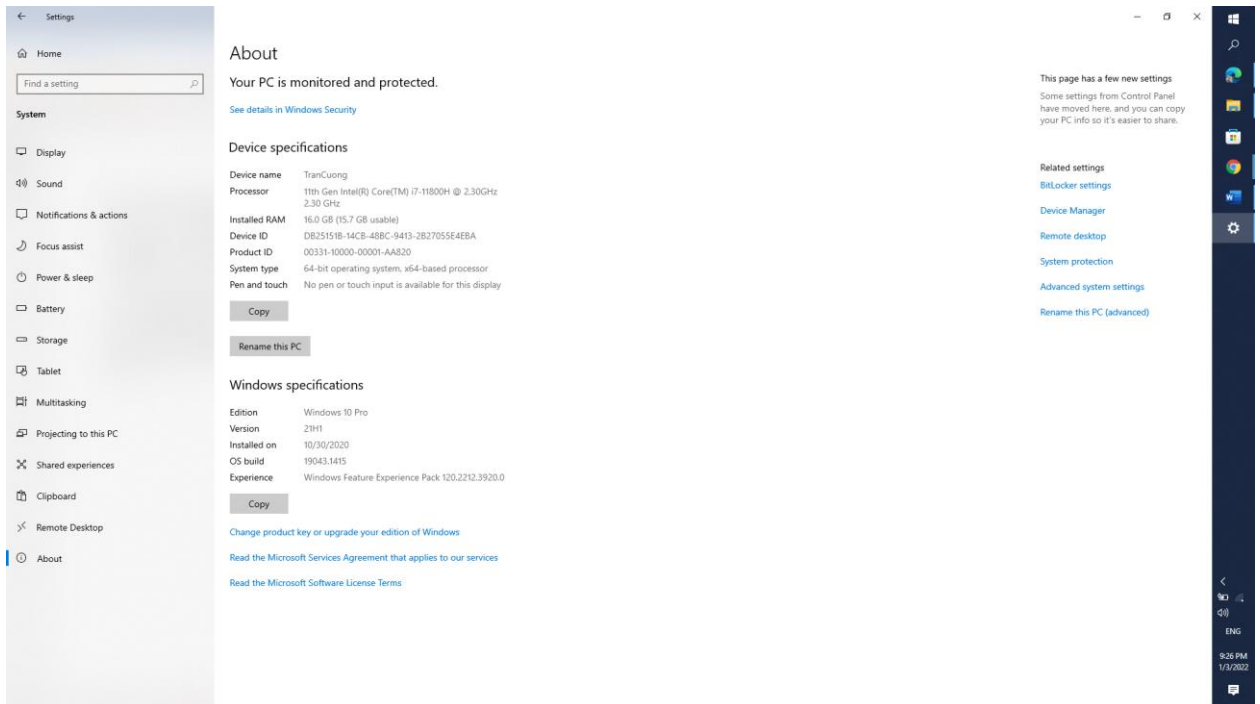


Ban đầu, hệ điều hành sẽ tìm một khung trống trong danh sách các khung còn trống. Trang tương ứng sẽ được đọc từ đĩa vào khung trang vừa tìm được. Sau đó, khoản mục tương ứng trong bảng phân trang sẽ được sửa đổi tương ứng với vị trí của trang trong bộ nhớ, lệnh thực hiện tiến trình đã ngắt sẽ được thực hiện lại với trang đã được nạp vào. Sau khi ngắt lệnh được xử lý xong, nếu gặp tiếp tục một trường hợp lỗi trang tương tự thì việc xử lý như trên tiếp tục được lặp lại.

II.1.3. Khảo sát các thông số về bộ nhớ trên máy tính cá nhân

II.1.3.1. Số bit quản lý địa chỉ ô nhớ

Thực hiện việc truy cập: Start → Settings → About, tra cứu ở phần Device specifications để biết được máy tính đang sử dụng hệ điều hành 32bit hay 64bit. Ở đây đang sử dụng hệ điều hành 64bit.



Với việc sử dụng hệ điều hành 64bit thì số bit để quản lý địa chỉ ô nhớ là 64 bit.

II.1.3.2. Kích thước trang

Thường thì kích thước trang đối với các hệ điều hành 32 hay 64bit hiện nay đang là 4KB, nhưng để chắc chắn hơn thì có một cách tra cứu thông số này.

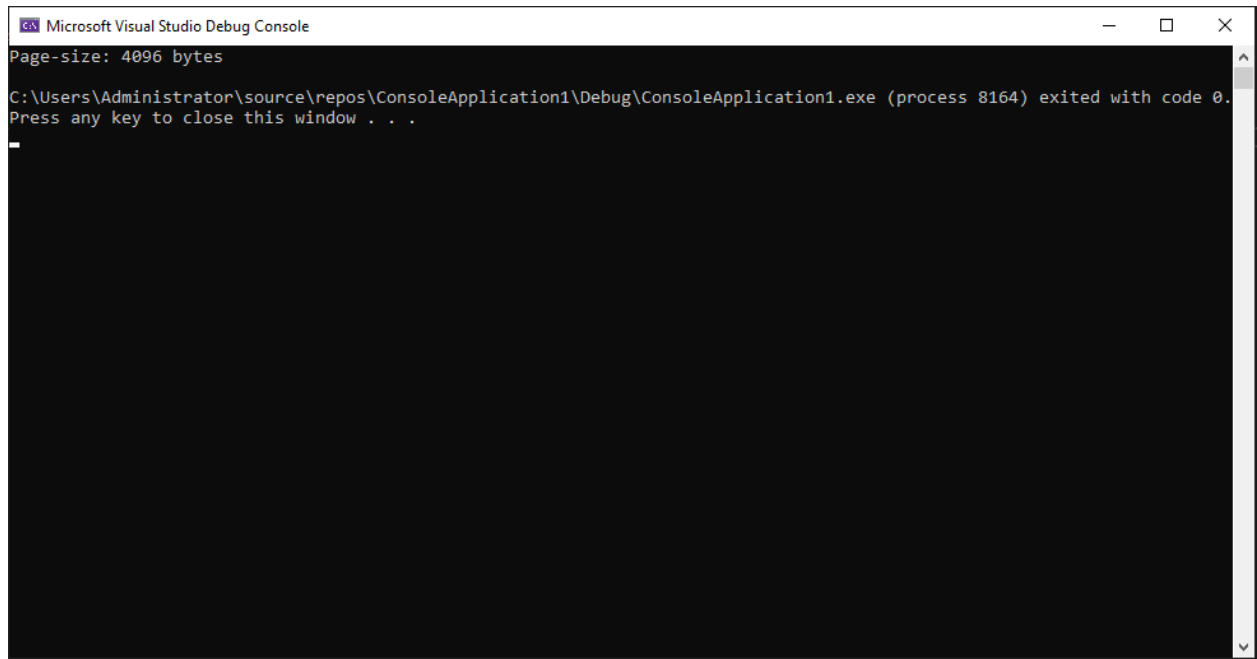
Bật Visual Studio 2019 (hoặc bất kỳ Editor hay IDE nào có hỗ trợ ngôn ngữ C++) và chạy chương trình với mã nguồn như sau:

```

1  #include <iostream>
2  #include <Windows.h>
3
4  using namespace std;
5
6  int main()
7  {
8      SYSTEM_INFO si;
9      GetSystemInfo(&si);
10
11     cout << "Page-size: " << si.dwPageSize << " bytes" << endl;
12
13     return 0;
14 }
15
16

```

Khi chạy chương trình, thông số pagesize của máy sẽ được hiển thị trên màn hình console:



```
Microsoft Visual Studio Debug Console
Page-size: 4096 bytes
C:\Users\Administrator\source\repos\ConsoleApplication1\Debug\ConsoleApplication1.exe (process 8164) exited with code 0.
Press any key to close this window . . .
```

Ở đây, giá trị trên máy tính cá nhân khảo sát được là 4096 byte hay 4KB.

II.1.3.3. Số bit tối thiểu để quản lý các offset trong trang

Là số bit cần dùng để mô tả tất cả các địa chỉ có trong một trang, được xác định như sau:

Kích thước trang là 4KB (đã khảo sát ở trên) = 4096 byte = 2^{12} byte, nghĩa là cần dùng tối thiểu 12 bit để quản lý tất cả các offset trong trang.

Vậy số bit tối thiểu để quản lý các offset trong trang là 12 bit (ở máy tính cá nhân có kích thước trang là 4KB)

II.1.3.4. Số khung trang vật lý

Được xác định bằng công thức: $\frac{\text{kích thước bộ nhớ vật lý}}{\text{kích thước trang}}$.

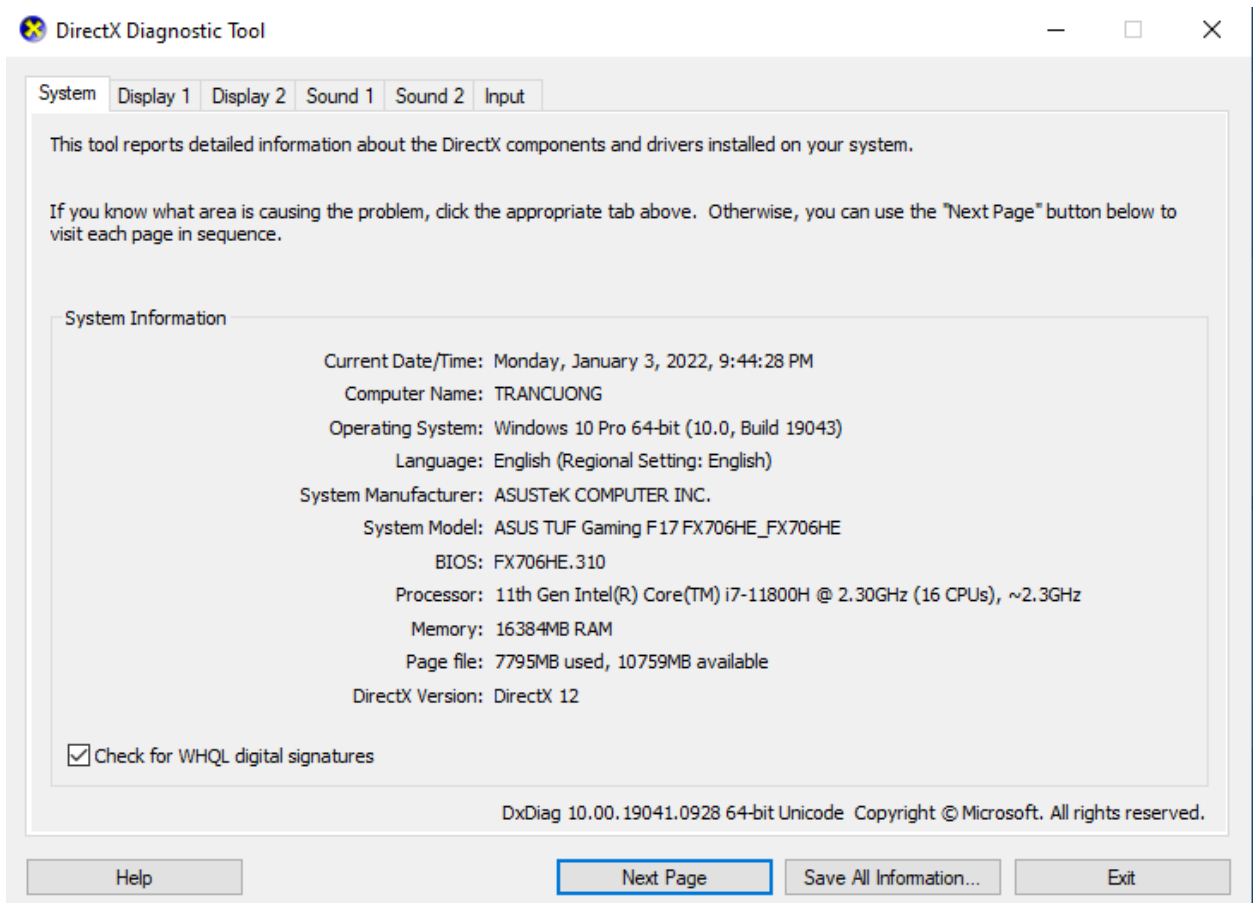
Ở đây, bộ nhớ vật lý hay bộ nhớ chính (mặc định là RAM), được khảo sát bằng cách dùng lệnh systeminfo trong cmd, số liệu ở đây là 16123MB (ở đây để dễ tính

toán ta tính tròn là 16GB, nếu khảo sát bằng lệnh dxdiag ở cửa sổ run)

```

Select Administrator: Command Prompt

BIOS Version: American Megatrends International, LLC. FX706HE.310, 11/25/2021
Windows Directory: C:\Windows
System Directory: C:\Windows\system32
Boot Device: \Device\HarddiskVolume1
System Locale: en-us;English (United States)
Input Locale: en-us;English (United States)
Time Zone: (UTC+07:00) Bangkok, Hanoi, Jakarta
Total Physical Memory: 16,123 MB
Available Physical Memory: 10,041 MB
Virtual Memory: Max Size: 18,555 MB
Virtual Memory: Available: 10,758 MB
Virtual Memory: In Use: 7,797 MB
Page File Location(s): C:\pagefile.sys
Domain: WORKGROUP
Logon Server: \\TRANCUONG
Hotfix(s): 10 Hotfix(s) Installed.
[01]: KB5007289
[02]: KB4562830
[03]: KB4570334
[04]: KB4577266
[05]: KB4577586
[06]: KB5000736
[07]: KB5008212
[08]: KB5006753
[09]: KB5007273
[10]: KB5005699
Network Card(s): 3 NIC(s) Installed.
[01]: Realtek PCIe GbE Family Controller
Connection Name: Ethernet
Status: Media disconnected
  
```



Kích thước trang đã được xác định là 4KB, do đó số khung trang vật lý là:

$$\frac{16384MB}{4KB} = \frac{16384 * 2^{10}KB}{4KB} = 4194304 \text{ (khung trang)}$$

II.1.3.5. Số khung trang logic tối đa trên không gian tiến trình

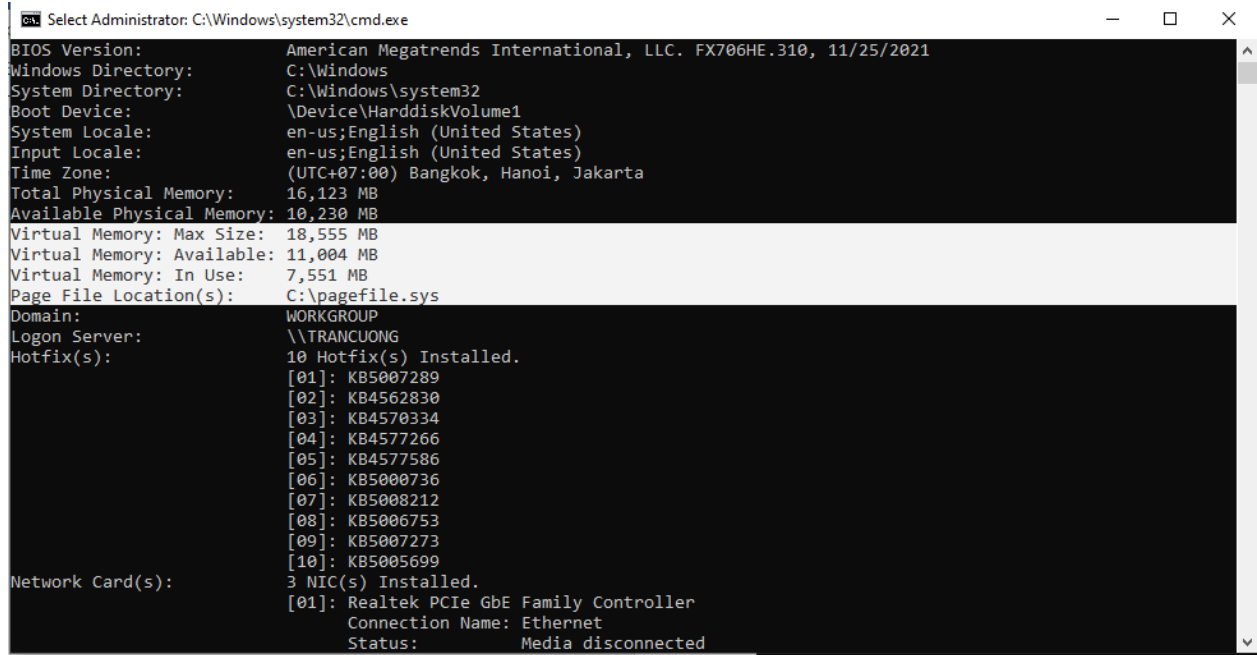
Được xác định bằng công thức: $\frac{\text{kích thước không gian tiến trình}}{\text{kích thước trang}}$.

Ta tính được:

$$\frac{2^{64}\text{byte}}{4\text{KB}} = \frac{2^{64}\text{byte}}{2^{12}\text{byte}} = 2^{52} \text{ (khung trang)}$$

II.1.3.6. Thông số về kích thước bộ nhớ ảo

Mở cmd lên, thực hiện lệnh systeminfo, ta sẽ tìm được các thông tin về bộ nhớ ảo (virtual memory):



```

Select Administrator: C:\Windows\system32\cmd.exe
BIOS Version: American Megatrends International, LLC. FX706HE.310, 11/25/2021
Windows Directory: C:\Windows
System Directory: C:\Windows\system32
Boot Device: \Device\HarddiskVolume1
System Locale: en-us;English (United States)
Input Locale: en-us;English (United States)
Time Zone: (UTC+07:00) Bangkok, Hanoi, Jakarta
Total Physical Memory: 16,123 MB
Available Physical Memory: 10,230 MB
Virtual Memory: Max Size: 18,555 MB
Virtual Memory: Available: 11,004 MB
Virtual Memory: In Use: 7,551 MB
Page File Location(s): C:\pagefile.sys
Domain: WORKGROUP
Logon Server: \\TRANCUONG
Hotfix(s): 10 Hotfix(s) Installed.
[01]: KB5007289
[02]: KB4562830
[03]: KB4570334
[04]: KB4577266
[05]: KB4577586
[06]: KB5000736
[07]: KB5008212
[08]: KB5006753
[09]: KB5007273
[10]: KB5005609
Network Card(s): 3 NIC(s) Installed.
[01]: Realtek PCIe GbE Family Controller
Connection Name: Ethernet
Status: Media disconnected
```

Các thông số khảo sát được trên máy tính cá nhân như:

- Kích thước lớn nhất: 18555MB
- Khả dụng: 11004MB
- Đang sử dụng: 7551MB

II.2. Câu 2

II.2.1. Thông tin sơ lược về chương trình

Các thông tin cơ bản về chương trình:

- Ngôn ngữ lập trình: Python
- Môi trường viết code:
- Môi trường thực thi: hệ điều hành Windows, chạy dưới dạng console application.

Giải thích ý nghĩa của output:

- Nếu gặp định dạng khung giờ có dạng (ví dụ mẫu): F06:00 T06:45, có nghĩa đây là thể hiện cho khung giờ từ 06:00 đến 06:45.
- Nếu gặp định dạng (ví dụ mẫu): F07:30 T11:30 D60 I20, có nghĩa là trong khoảng thời gian từ 07:30 đến 11:30 có thể sử dụng máy, nhưng mỗi lần bật

máy thì chỉ được dùng tối đa 60 phút – sau đó máy sẽ không hoạt động cho đến khi đã ngắt đủ 20 phút.

Để thuận tiện cho việc báo cáo, nhóm đã thay đổi tất cả các thời gian chờ và chạy thành 1 phút.

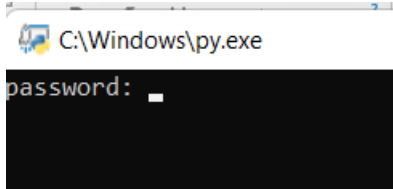
II.2.2. Chương trình C

Theo yêu cầu, chương trình sẽ được chạy ở chế độ autorun. Để thực hiện việc này, ta cần đặt file .py chứa source code của chương trình vào folder Startup bằng cách nhấn tổ hợp phím Windows + R (mở hộp thoại Run) sau đó nhập vào shell:startup)

Mô tả cách chương trình hoạt động:

Bước 1:

Khi khởi động máy tính, chương trình C sẽ được tự động chạy, việc đầu tiên là yêu cầu người sử dụng nhập mật khẩu:



Source code:

```
def inputPass():
    count = 0
    password = link_parent + r'\\password.txt'
    file = open(password, 'r')
    pass_ = file.readlines()
    while True:
        password = input('password: ')

        if password == pass_[1].strip():
            print('You input Child password')
            return "C"
        elif password == pass_[0].strip():
            print('You input Parent password')
            return "P"
        else:
            print('Incorrect password!')
            count += 1
            if count == 3:
                Time_can_use()
                Shut_down()
            return None
```

Khi nhận được mật khẩu do người dùng nhập vào, chương trình sẽ truy cập vào thư mục *sharing* với máy của phụ huynh. Tại đó, chương trình sẽ kiểm tra mật khẩu mà người dùng nhập vào. Nếu mật khẩu đó là parent thì sẽ trả về P, children thì trả về C còn nếu không thuộc 2 mật khẩu trên thì in ra “Incorrect Password” và

bắt người dùng nhập lại nếu nhập không đúng 3 lần thì sẽ tắt máy và khóa máy trong 10 phút tiếp theo không được sử dụng máy nhờ vào hàm Time_can_use.

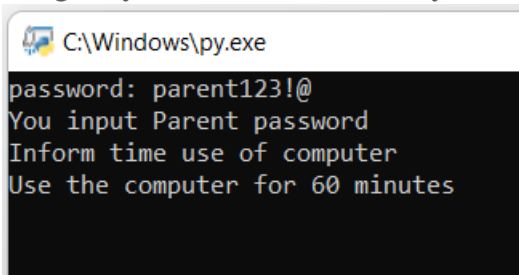
```
def Time_can_use(n = 10):  
    time_can_use = link_child + r'\\time_use.txt.txt'  
    with open(time_can_use, 'w') as f:  
        t = getTimeNow()  
        hour = int(t[:2])  
        minute = int(t[3:]) + n  
        if minute >= 60:  
            hour += 1  
            minute -= 60  
        f.write('{}:{}'.format(hour, minute))
```

Điều này sẽ lưu khung giờ sử dụng máy vào một file riêng và chương trình sẽ kiểm tra file này mỗi khi bật máy để xem người sử dụng đã có thể ở trong khung giờ được sử dụng máy hay chưa.

```
def main():  
    time_use = link_child + '\\time_use.txt'  
    with open(time_use, 'r') as f:  
        time_wait = f.readline().strip()  
        n = hour_to_num(getTimeNow()) - hour_to_num(time_wait)  
        if n > 0:  
            f.close()  
        else:  
            speech = pyttsx3.init()  
            inform = '{} minutes left until using computer time.'.format(n)  
            speech.say(inform)  
            speech.runAndWait()  
            f.close()  
            Shut_down()
```

Bước 2:

Trường hợp mật khẩu đã nhập là của phụ huynh, chương trình sẽ in ra thông tin cho biết mật khẩu vừa rồi là của phụ huynh và bạn có 60 phút đếm ngược để sử dụng máy, tính từ thời điểm này.



```
C:\Windows\py.exe  
password: parent123!@  
You input Parent password  
Inform time use of computer  
Use the computer for 60 minutes
```

Hết 60 phút, chương trình sẽ quay trở lại bước 1 và người sử dụng phải nhập lại mật khẩu một lần nữa.

```
C:\Windows\py.exe
password: parent123!@
You input Parent password
Inform time use of computer
Use the computer for 60 minutes
password:
```

Bước 3:

Trường hợp mật khẩu đã nhập là của trẻ, chương trình sẽ kiểm tra rằng trẻ đã ở trong khung thời gian được sử dụng máy hay chưa.

Nếu hiện tại không nằm trong khung thời gian được sử dụng máy, chương trình sẽ thông báo ra khoảng thời gian tiếp theo sử dụng máy và mở 1 tiến trình tự tắt máy sau 15 giây và yêu cầu nhập mật khẩu, nếu người dùng nhập đúng mật khẩu của parent thì sẽ được sử dụng máy tiếp.

```
if check_in_time == False: #Cờ kiểm tra xem có nằm trong khung thời gian được sử dụng hay không
    inform_next_use(list_time)
    flag = False
    shut_down = Timer(15, Shut_down) #Create thread
    shut_down.start()
    while True:
        c = inputPass()
        if c == "P":
            flag = True
            shut_down.cancel() #Nếu nhập đúng mật khẩu của parent thì hủy lệnh Shutdown
            break
```

Nếu đang trong khung thời gian có thể sử dụng máy, chương trình sẽ in ra người dùng đã nhập mật khẩu trẻ và các khung thời gian sử dụng và in ra thời gian còn lại trẻ được dùng máy tính và sau bao lâu mới được mở lại lần nữa.

```
C:\Windows\py.exe
password: children111!@
You input Child password
From: 20:57
To: 21:30
Inform time use of Children
Shut down after 33 minutes
7 hours and 48 minutes fo the next using computer time.

Inform time use of computer
Use the computer for 33 minutes
_
```

Lúc này, chương trình sẽ thực hiện giám sát:

- Chạy chương trình chụp màn hình, lưu lại file chụp màn hình dựa theo ngày tháng năm giờ phút giây chụp vào thẳng thư mục sharing.


```
def takeScreenshot():
    filename = f'{datetime.now().strftime("%Y-%m-%d-%H-%M-%S")}.png'
    take_screen = pyautogui.screenshot()
    take_screen.save(r'C:\Users\ADMIN\OneDrive\TinTran\data_child\ScreenShot\{}'.format(filename))
```

Lệnh trong hàm main này dùng để tạo 1 tiến trình riêng sẽ chụp màn hình mỗi 60 giây vừa qua:

```
take_scr = RepeatTimer(60, takeScreenshot)
```

- Chạy chương trình kiểm tra thay đổi trong file của phụ huynh. Hàm checkDiff kiểm tra dữ liệu trong file đã bị thay đổi hay chưa, nếu đã bị thay đổi thì chương trình sẽ cập nhật lại các khung thời gian mới.

```
def checkDiff(filename = r'C:\Users\ADMIN\OneDrive\TinTran\data\time.txt', list_time = list_time):
    list_new = readfile_time(filename)
    if list_new != list_time:
        print('File have been changed')
        for i in range(len(list_time)):
            list_time[i] = list_new[i]
```

Lệnh dưới đây được sử dụng để tạo 1 tiến trình kiểm tra file đã bị thay đổi hay chưa cứ 60 giây 1 lần

```
checkfile_change = RepeatTimer(60, checkDiff)
```

- Chương trình thông báo khi chỉ còn 1 phút sử dụng
Hàm checkTime có nhiệm vụ thông báo:

```
def checkTime():
    speakOut("You have 1 minutes left using the computer")
```

Hàm speakOut dùng để thông báo qua loa:

```
def speakOut(string):
    speech = pyttsx3.init()
    speech.say(string)
    speech.runAndWait()
```

Sử dụng 2 dòng lệnh sau để tạo 1 tiến trình thông báo khi chỉ còn 1 phút

```
count_time = Timer((t_duration - 1)*60, checkTime)
```

```
count_time = Timer(hour_to_num(t_to) - 1 - hour_to_num(getTimeNow())*60, checkTime)
```

II.2.3. Chương trình P

Chương trình P được tạo ra với các chức năng sau:

II.2.3.1. Xem thông tin thời khóa biểu

Hàm xem và in ra thời khóa biểu được sử dụng máy của trẻ:

```
# Xem thời khóa biểu của trẻ
def viewSchedule(filename):
    print('\n=====THOI KHOA BIEU=====\\n')
    with open(filename, 'r') as file:
        for i, line in enumerate(file.readlines()):
            print('Day {} - {}'.format(i + 1, line))
```

Cần có một hàm để đọc ra các thông tin có trong file và trả về danh sách các giá trị thời gian bắt đầu học, thời gian kết thúc, thời lượng mỗi lần truy cập, thời gian ngắt:

```
# Đọc và lấy các giá trị từ file
def getDataFromFile(filename):
    with open(filename, 'r') as file:
        time_from_list = []
        time_to_list = []
        duration_list = []
        interupt_list = []

        for i, line in enumerate(file.readlines()):
            line = line.strip().split()
            signal = line[0].find(':')
            time_from_list.append(
                Time(int(line[0][1:signal]), int(line[0][signal+1:])))
            signal = line[1].find(':')
            time_to_list.append(
                Time(int(line[1][1:signal]), int(line[1][signal+1:])))

            if len(line) > 2:
                for index, temp in enumerate(line):
                    if index > 1:
                        if temp[0] == 'D':
                            duration_list.append((i, int(temp[1:])))
                        elif temp[0] == 'I':
                            interupt_list.append((i, int(temp[1:])))

        return time_from_list, time_to_list, duration_list, interupt_list
```

Output của chức năng này:

```
=====THOI KHOA BIEU=====
```

```
Day 1 - F06:00 T07:45 D60 I10
```

```
Day 2 - F07:30 T08:30 D60 I25
```

```
Day 3 - F19:00 T21:30
```

II.2.3.2. *Chỉnh sửa, điều chỉnh file*

Vì chương trình P có thể được truy cập từ cả 2 máy (máy cha và máy mẹ) nên ở chức năng này cần phải xử lý bài toán tránh 2 tiến trình từ 2 máy cùng đi vào critical section. Ở đây, nhóm sử dụng semaphore để giải quyết bài toán này:

Bước 1:

Đọc giá trị semaphore từ file **flag.txt** (1: cho phép tiến trình vào xử lý, 0: tiến trình phải đợi do có tiến trình khác đang xử lý)

```
with open(flagfile, 'r') as file:
    semaphore = int(file.readline())
```

Bước 2:

Kiểm tra giá trị semaphore, nếu khác 1 thì có nghĩa tiến trình cần phải đợi và quay lại để tiếp tục việc kiểm tra:

```
# Nếu semaphore = 0 có nghĩa là đang có một tiến trình khác xử lý
if semaphore != 1:
    time.sleep(2)
    continue
```

Nếu bằng 1 thì sẽ tạo 1 đối tượng *semaphore* với giá trị đầu vào là giá trị semaphore đọc được:

```
semaphore = threading.Semaphore(semaphore)
```

Bước 3:

Đặt lại giá trị semaphore trong file **flag.txt** để ngăn các tiến trình khác đi vào

```
# Giảm counter xuống 1
semaphore.acquire()

#Viết vào file flag.txt giá trị semaphore mới để biết được đang có tiến trình xử lý
with open(flagfile, 'w') as file:
    file.write(str(semaphore._value))
```

Bước 4:

Thực hiện việc điều chỉnh và cập nhật lại file thời khoá biểu của trẻ

```
#Thực hiện việc điều chỉnh file thời khoá biểu của trẻ
time_from_list, time_to_list, duration_list, interrupt_list = getDataFromFile(
    filename)
configTimeSchedule(time_from_list, time_to_list,
                    duration_list, interrupt_list)
writeToFile(filename, time_from_list, time_to_list,
            duration_list, interrupt_list)
```

Bước 5:

Đặt lại giá trị semaphore lên 1 và cập nhật lại file **flag.txt** để đánh dấu đã xong và tiến trình khác có thể truy cập vào, sau đó thoát chương trình

```
#Tăng counter lên 1
semaphore.release()

#Set lại giá trị semaphore để biết đã hoàn thành và trả lại cho các tiến trình khác đi vào xử lý
with open(flagfile, 'w') as file:
    file.write(str(semaphore._value))
print(semaphore)
break
```

Output của chức năng này sẽ được nhóm thể hiện trong video demo.

II.2.3.3. Xem lịch sử sử dụng máy tính của trẻ

```
# Xem lịch sử đăng nhập vào máy của trẻ
def viewHistoryAction(filename):
    print('\n=====LICH SU DANG NHAP TREN MAY CUA TRE=====\\n')
    with open(filename, 'r') as file:
        for line in file.readlines():
            print(line)
```

Output chức năng này sẽ in ra lịch sử đăng nhập trên máy trẻ để ba mẹ có thể quản lý với mỗi dòng tương ứng là ngày tháng năm, thời gian đăng nhập trên máy

=====LICH SU DANG NHAP TREN MAY CUA TRE=====

12:01:2022, 16:06 - 16:06

12:01:2022, 16:09 - 16:09

12:01:2022, 16:14 - 17:00

12:01:2022, 16:14 - 17:00

12:01:2022, 16:14 - 17:00

12:01:2022, 16:14 - 17:00

12:01:2022, 16:14 - 17:00

12:01:2022, 16:14 - 17:00

12:01:2022, 16:14 - 17:00

12:01:2022, 16:14 - 17:00

12:01:2022, 16:17 - 17:00

12:01:2022, 16:18 - 17:00

12:01:2022, 16:21 - 17:00

12:01:2022, 16:22 - 17:00

III. Tham khảo

Chung:

- Slide bài giảng của thầy Thái Hùng Văn, trường Đại học Khoa Học Tự Nhiên.

Câu 1:

- Sách Operating System Concepts của các tác giả: Abraham Silberschats; Peter Baer Galvin, Grey Gagne
- Tham khảo một ít từ các nguồn tài liệu ngẫu nhiên trên Internet.

Câu 2:

- <https://www.geeksforgeeks.org/synchronization-by-using-semaphore-in-python/>
- https://stackoverflow.com/questions/12435211/python-threading-timer-repeat-function-every-n-seconds?fbclid=IwAR1RNnOz24ajsvc7gqqA1Ln-JI6_1JHkD7imd2ucIcRB4CGtQIWQ4ziHZ9g