**N7-Nguyễn Hoàng Dương-B19DCCN153**

**Bài kiểm tra cuối kỳ, môn Hệ Điều Hành.**

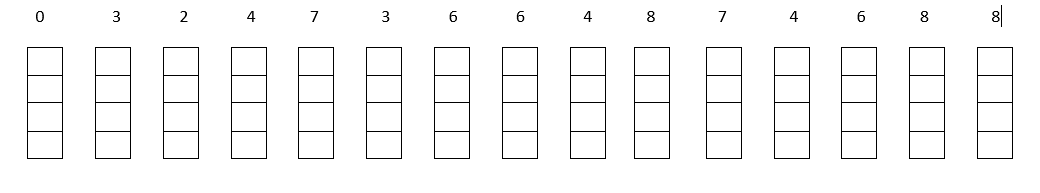
**Học phần: Hệ điều hành (Học kỳ 1 2021-2022)**

**Thời gian làm bài: 90 phút**

**Đề số 1**

**Câu 1 (3 điểm):**

Một tiến trình được cấp 4 khung để nạp code. Thứ tự các trang nạp theo yêu cầu có trình tự như trong hình. Hãy điền số trang vào các ô khung khi cần nạp trang nếu xẩy ra tình trạng thiếu khung cho trang bằng **phương pháp Tối ưu (OPT)**

****

**Câu 2 (3 điểm):**

1. Trình bày phân chương động bộ nhớ. Phân tích ưu nhược điểm của phương pháp này so với phân chương cố định.
2. Cho ví dụ minh họa cấp phát động với các chiến thuật: **firrst fit, best fit, worst fit**.
3. Khi di chuyển chương sang vị trí khác, cần thay đổi thông tin gì trong khối ánh xạ địa chỉ.

**Câu 3 (4 điểm):**

1. Trình bày khái niệm, tổ chức khi phân trang bộ nhớ.
2. Trình bày thuật ánh xạ địa chỉ khi phân trang bộ nhớ.
3. Không gian bộ nhớ logic của TT gồm có 4 trang, mỗi trang có kích thước 4096 bytes. Bộ nhớ vật lí có 64 khung. Để biểu diễn địa chỉ logic và địa chỉ vật lí cần bao nhiêu bits ?
4. **Cho một địa chỉ logic= 12046. Hãy tính địa chỉ vật lí của địa chỉ logic đó**

**dựa vào bảng trang sau:**

|  |  |
| --- | --- |
| Số trang | Số khung |
| 0 | 13 |
| 1 | 10 |
| 2 | 15 |
| 3 | 5 |

**Câu 4 (4 điểm): (Điểm tham gia thực hành có mặt 2 buổi: 6 đ và bài tập này cộng lại)**

Cho trước một hàm addcount( ) tính giá trị biến counter, với số vòng lặp chứa trong biến loops. Viết chương trình main() nhận từ bàn phím số đếm cho biến loops. Trong main tạo 2 luồng cùng dùng chung hàm addcount().Chạy thử với các số đếm tăng dần tới khi thấy kết quả sai như dự đoán. Sau đó thực hiện đồng bộ thực thi bằng cách loại trừ lẫn nhau (MUTEX) để đảm bảo kết quả đứng. Sử dụng các hàm: **pthread\_mutex\_lock(&m);, pthread\_mutex\_unlock(&m); pthread\_exit(NULL);**

**(Cho trước các dữ kiện sau: )**

#include <stdlib.h>

#include "common.h"

#include <pthread.h>

pthread\_mutex\_t m; //khai báo đối tượng MUTEX

volatile int counter = 0;

int loops;

void \*addcount(void \*arg)

{

int i;

for (i = 0; i < loops; i++)

{ counter = counter + 1;}

}

**main() { ….. }**

**BÀI LÀM**

**Câu 1:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **3** | **2** | **4** | **7** | **3** | **6** | **6** | **4** | **8** | **7** | **4** | **6** | **8** | **8** |
| 0 | 0 | 0 | 0 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 |
|  | 3 | 3 | 3 | 3 | 3 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 |
|  |  | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 8 | 8 | 8 | 8 | 8 | 8 |
|  |  |  | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |

**Câu 2:**

* Kỹ thuật phân chương động:
* Kích thước, số lượng và vị trí của chương đề có thể thay đổi
* Mỗi một tiến trình được cấp 1 chương nhớ có kích thước đúng bằng chương đó, khi tiến trình kết thúc, chương nhớ được giải phóng tạo thành 1 vùng trống. Các vùng trống nằm liền nhau được gộp thành 1 vùng trống lớn hơn. Các chương nhớ và các vùng trống được liên kết với nhau thành 1 danh sách kết nối
* Ưu nhược điểm:
* Ưu điểm: Tránh phân mảnh trong do các trường được tạo ra đúng bằng kích thước tiến trình
* Nhược điểm: Gây phân mảnh ngoài: dồn những vùng trống nhỏ thành lớn. Làm lãng phí bộ nhớ.

1. Ví dụ:

Trong bộ nhớ có 4 vùng trống có kích lước như sau 10MB, 20MB, 18MB,24MB; yêu cầu cấp phát vùng nhớ kích thước 16MB.

Khi đó thì ba chiến lược cấp phát cho kết quả:

* Chiến lược first fit: chọn khối 20MB để chia và cấp phát
* Chiến lược best fit: chọn vùng trống 18MB
* Chiến lược worst fit: chọn vùng trống 24MB

1. Nếu chương bị di chuyển thì nội dung của thanh ghi cơ sở bị thay đổi chứa địa chỉ vị trí mới

**Câu 3:**

* Khái niệm phân trang bộ nhớ:
* Bộ nhớ vât lý được chia thành các khối nhỏ, có kích thước cố định và bằng nhau gọi là khung trang (page frame)
* Không gian địa chỉ logic của tiến trình được chia thành những khối gọi là trang (page), có kích thước bằng khung
* Tiến trình được cấp các khung để chứa các trang của mình, các trang có thể chứa trong các khung nằm rải rác trong bộ nhớ
* Ưu nhược điểm:
* Ưu điểm: không phân mảnh ngoài
* Nhược điểm: bỏ phí trang do phân mảnh trong

1. Ánh xạ địa chỉ khi phân trang bộ nhớ:

* Để tính toán địa chỉ hiệu quả, kích thước khung được chọn là lũy thừa của 2
* Địa chỉ logic gồm 2 phần:
* Số thứ tự trang (p)
* Độ dịch (địa chỉ lệch) của địa chỉ so với đầu trang (o)
* Nếu kích thước trang là 2n, số trang 2m. Biểu diễn địa chỉ logic dưới dạng địa chỉ có độ dài (m + n) bit
* m bit cao: biểu diễn số thứ tự trang
* n bit thấp: biểu diễn độ dịch trong trang nhớ
* Quá trình chuyển địa chỉ logic sang địa chỉ vật lý:
* Lấy m bit cao của địa chỉ => được số thứ tự trang
* Dựa vào bảng trang, tìm được số thứ tự khung vật lý (k)
* Địa chỉ vật lý bắt đầu của khung là k\*2n
* Địa chỉ vật lý của byte được tham chiếu là địa chỉ bắt đầu của khung cộng với địa chỉ lệch (độ dịch)
* Chỉ cần thêm số khung vào trước dãy bit biểu diễn độ lệch

1. **Tính số bits cần:**

Theo bảng, có 4 (trang) = 22 => dùng 2 bit biểu diễn trang.

Kích thước 1 trang: 4096 = 212 => dùng 12 bit biểu diễn độ dịch.

* Cần 2+12 = 14 bit để biểu diễn địa chỉ logic.

Lại có: Bộ nhớ vật lý: 64 khung = 26 => dùng 6 bit biểu diễn khung.

* Cần 6+12 = 18 bit để biểu diễn địa chỉ vật lý.

1. **Tính địa chỉ vật lý:**

Kích thước khung bộ nhớ 4096 byte = 2^12

Địa chỉ logic 12046, ta có:

p = 12046/4096 = 2 -> Số thứ tự trang = 2

o = 12046 mod 4096 = 3854

Theo bảng trang, trang 2 nằm ở số khung trang 15.

* + Địa chỉ vật lý tương ứng là 15\*4096 + 3854 = 65294

**Câu 4:**

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include "common.h"

pthread\_mutex\_t m;

volatile int counter = 0;

int loops;

void \*addcount(void \*arg)

{

pthread\_mutex\_lock(&m);

int i;

for (i = 0; i < loops; i++) {

counter = counter + 1;

printf("thread %d - counter: %d\n", arg, counter);

}}

main() {

pthread\_t threads[2];

scanf("%d", &loops);

pthread\_create(&threads[0], NULL, &addcount, (int \*)0);

pthread\_create(&threads[1], NULL, &addcount, (int \*)1);

}