1. **What is a process control block(PCB)? Summarize the typical organization and content of a PCB.**

课件上并没有明确到底什么是PCB，只有PCB是来干什么的，下面两段是课件上的：

For each process, the operating system maintains a process control block (PCB) or process descriptor to **keep a clear picture of what each process is doing**. what point it has reached in its execution and what resources have been assigned to it.

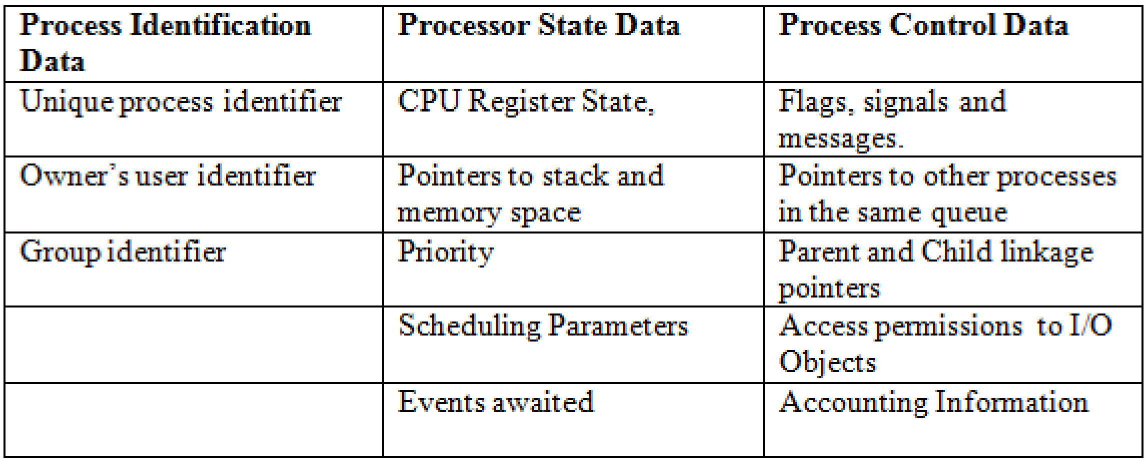
A Process Control Block is used to **keep track of the execution context** (all resource information about the process and its activity) that can be used for independent scheduling of that process onto any available processor.

(百度）为了描述控制进程的运行，系统中存放进程的管理和控制信息的数据结构称为进程控制块

(Baidu)In order to describe the operation of the control process, the data structure storing the management and control information of the process in the system is called the process control block.

Organization and content of a PCB：

Process Identification、Data Processor State、Process Control Data



Process Identification Data	**Processor State Data**	**Process Control Data**
Unique process identifier	CPU Register State,	Flags, signals and messages.
Owner's user identifier	Pointers to stack and memory space	Pointers to other processes in the same queue
Group identifier	Priority	Parent and Child linkage pointers
	Scheduling Parameters	Access permissions to I/O Objects
	Events awaited	Accounting Information

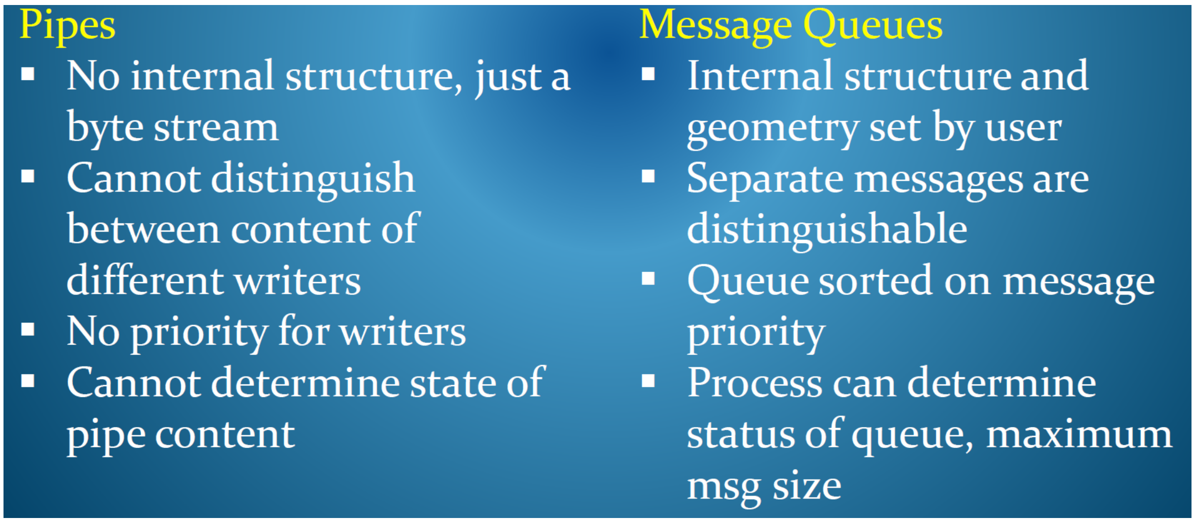
2. **Explain clearly how a process is created in Unix and how the new child process distinguishes itself from the parent and can follow a different execution sequence while retaining access to its parent's I/O devices.**

A process is created using the `fork()` system call within C program. **Initial process is the Parent and new process is the Child.**

we could distinguish child process from parent process by **pid**. **The pid of child is 0 and parent is bigger than 0;**

The child process could follow a different execution sequence by `exec()` while retaining access to parent's IO devices

3. **Compare and contrast the two Unix interprocess communication mechanisms:- message queues and pipes.**



Pipes

- No internal structure, just a byte stream
- Cannot distinguish between content of different writers
- No priority for writers
- Cannot determine state of pipe content

Message Queues

- Internal structure and geometry set by user
- Separate messages are distinguishable
- Queue sorted on message priority
- Process can determine status of queue, maximum msg size

4. **In disk scheduling, explain what advantages the C-LOOK algorithm has over the SCAN algorithm? Support your answer with an example.**

C-LOOK checks to see if there are anymore requests in that direction before flying back. SCAN need to go to the last track in one direction before turn around; C-LOOK will be more efficient than SCAN.

Example: 98,183,37,122,14,124,65,67(you are at 53)

C-LOOK: 53,65,67,98,122,124,187,fly back,0,14,37=...

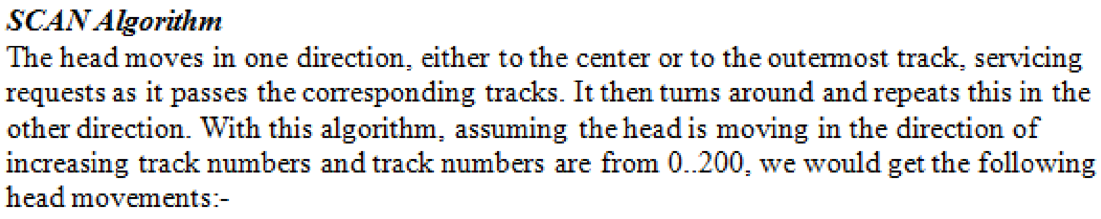
SCAN: 53,65,67,98,122,124,187,200,37,14=...

肯定是C-LOOK > SCAN

资料:

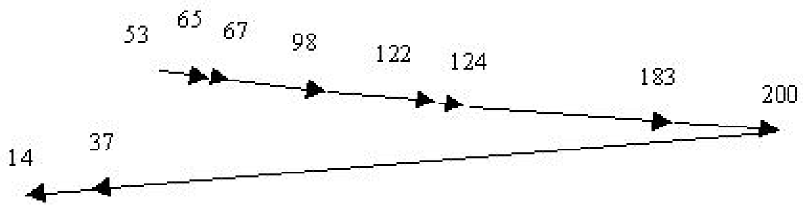
(1)前缀C表示是否会飞回flyback(飞回)，可以认为flyback非常的快，相同点：C他们都是按照一个方向，飞回都是飞到最内或者最外侧(0)

(2)SCAN与LOOK的区别就在于SCAN是扫描所有的track，一定要到最外面或者最里面；LOOK只到我们需要的最内层或者最外层

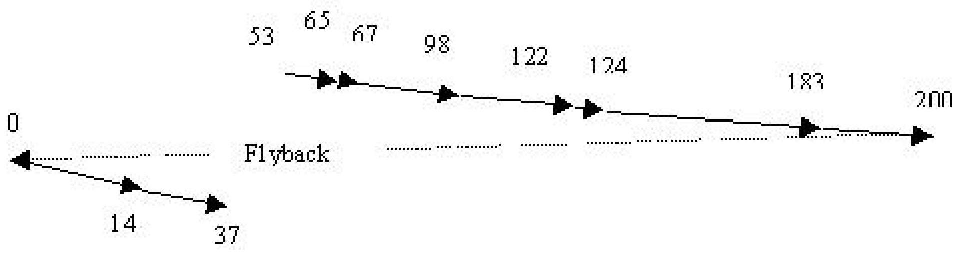


SCAN Algorithm

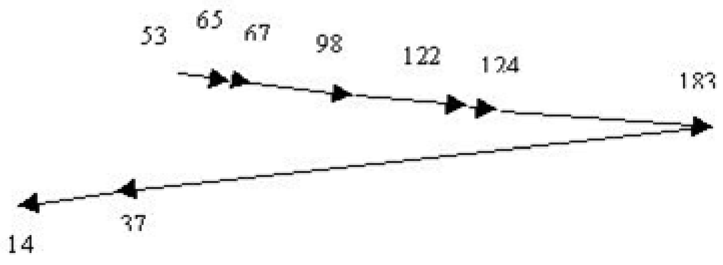
The head moves in one direction, either to the center or to the outermost track, servicing requests as it passes the corresponding tracks. It then turns around and repeats this in the other direction. With this algorithm, assuming the head is moving in the direction of increasing track numbers and track numbers are from 0..200, we would get the following head movements:-



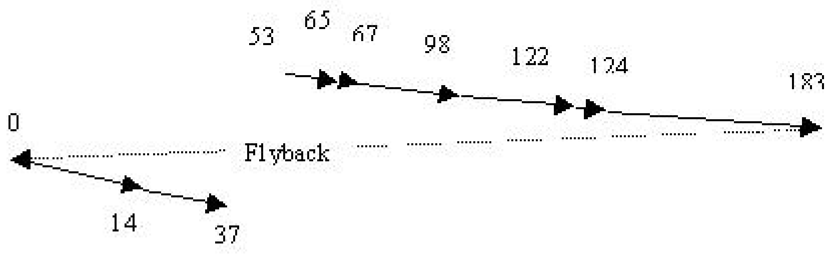
C-SCAN Algorithm
This algorithm attempts to regularise the distribution of service offered by SCAN. When the head has completed movement in one direction, it flys back quickly to the start, without scaning the surface, and begins the scan again in the same direction. Each track then has the same time between consecutive visits by the head.
With this algorithm, we get the following head movements:-



LOOK Algorithm (Elevator Algorithm)
The LOOK algorithm is the same as SCAN except that where SCAN always continues to the last track in a particular direction before turning around, the LOOK algorithm simply checks to see if there are anymore requests in that direction before turning around.
With this algorithm we get the following head movements:-



C-LOOK Algorithm
The C-LOOK algorithm is the same as C-SCAN except that where C-SCAN always continues to the last track in a particular direction before flying back, the C-LOOK algorithm simply checks to see if there are anymore requests in that direction before flying back.
With this algorithm we get the following head movements:-

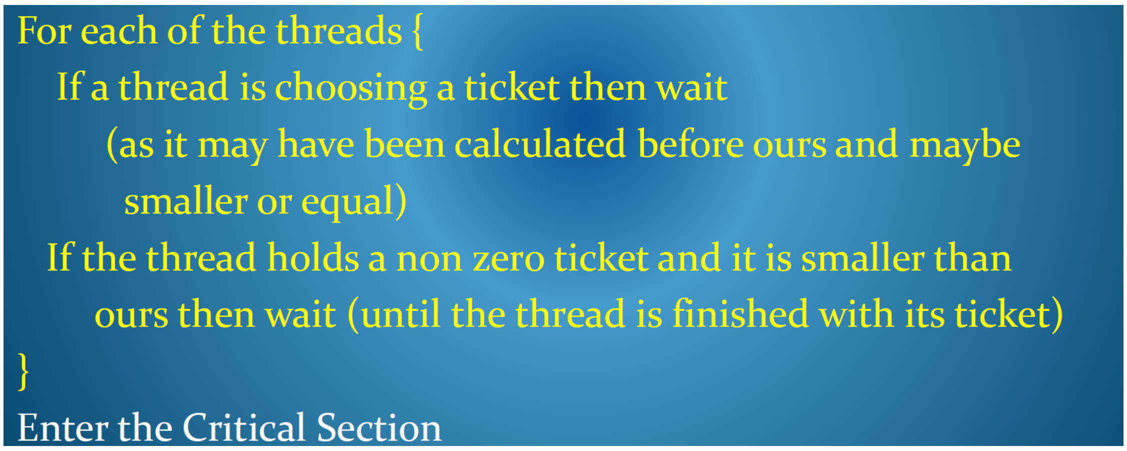


5. **Give a psuedo code software solution to the n-process mutual exclusion problem indicating the entry code and exit code to be executed by each process. Explain the components of your code.**

进入前无非就是两个动作：

1、摇一个号（表示按照这个号码排队进入面包店）

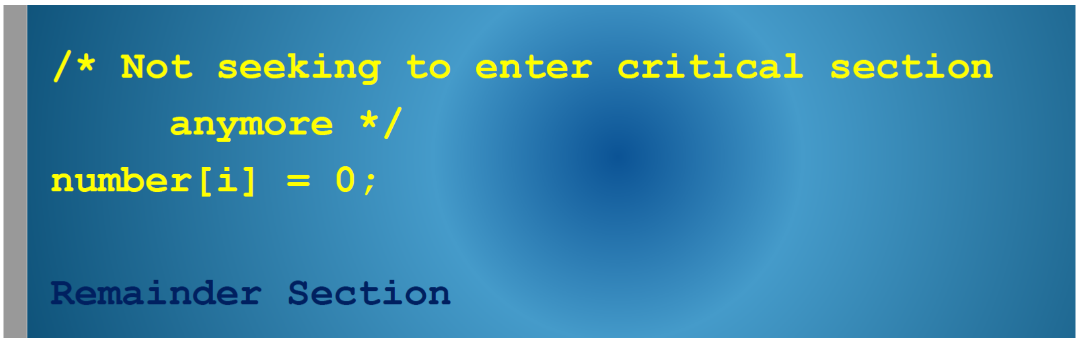
2、判断线程有号并且这个号是最小的（有排队进面包店的号码，并且比你小的都已经进去了，现在到你进去了）



```
For each of the threads {
    If a thread is choosing a ticket then wait
         (as it may have been calculated before ours and maybe
           smaller or equal)
   If the thread holds a non zero ticket and it is smaller than
        ours then wait (until the thread is finished with its ticket)
}
Enter the Critical Section
```

出去的时候只需要一个动作：

把我原来摇到的号置成0，表明现在没有在排队



```
/* Not seeking to enter critical section
     anymore */
number[i] = 0;

Remainder Section
```

一定要理解面包房算法的原则：

定义(理解这个非常关键！！！)

- 数组choosing[i]为真，表示进程i正在获取它的排队登记号；
- 数组number[i]的值，是进程i的当前排队登记号。如果值为0，表示进程i未参加排队，不想获得该资源。规定这个数组元素的取值没有上界。
- 正在访问临界区的进程如果失败，规定它进入非临界区，number[i]的值置0，即不影响其它进程访问这个互斥资源。

```
//排队的数组，初始值为false；false表示不在排队摇号，true表示在排队摇号
boolean[] choosing;
//登陆号码的数组，初始值为0，记录排队的登记号码，这个号码决定了进入的先后顺序
int[] number;
//下面是所谓的“摇号”
//开始摇号
choosing[i] = true;
//下面这句其实就是在这个已有的登陆号码数组中找到最大的那个，然后+1给这个正在摇号的线程
number[i]=Max(number[0],number[1], ...,number[n-1])+1;
//摇号结束
choosing[i] = false;
//线程一次等待，一个一个进入
for (j = 0; j<n; j++) {
//下面这句代码choosing[i]如果是true的话，就是表明正在排队摇号，那就要一直循环等待，直至摇完了号
while choosing[j] {}
//如果这个摇到的号不是0，就说明在排队，0说明不在排队，只要去摇号了，号就不会是0
//第二个要求其实是在说如果在同一时间摇号摇到的号一样，就要根据下标进行比较
while ((number[j] != 0) && ((number[j],j) < (number[i],i)))
{
/* Wait if a smaller ticket exists */
//等到一个可以进去的线程
}
}
//这个地方就是要进行操作的部分
/* Not seeking to enter critical section anymore */
//使用完了之后线程的号又变成了0，要是想再进去请重新排队
number[i] = 0;
Remainder Section
```

6. **State the Dining Philosophers problem and outline a deadlock free solution using semaphores.**

Let's say that we have a table on which there are five plates and five chopsticks with a bowl of rice in the centre. Every philosopher could do only two things think and eat, when he want to eat he must pick up the chopstick on his left and then pick up the right one. when all five philosophers pick up the left chopsticks, they cannot pick up the right one, and they are all in hungry for all time.

使用信号量解决哲学家就餐问题的方法有两种，这里选择一个我觉得比较好记的方法：设置信号量位4

Allow at most four philosophers to sit at the table at the same time. This means at least one philosopher can always eat and
when she's finished, another can eat and so on. We set the semaphores=4 and that mean when all 4 philosophers want to eat, there is always one could pick up a pair of chopsticks and then release them.

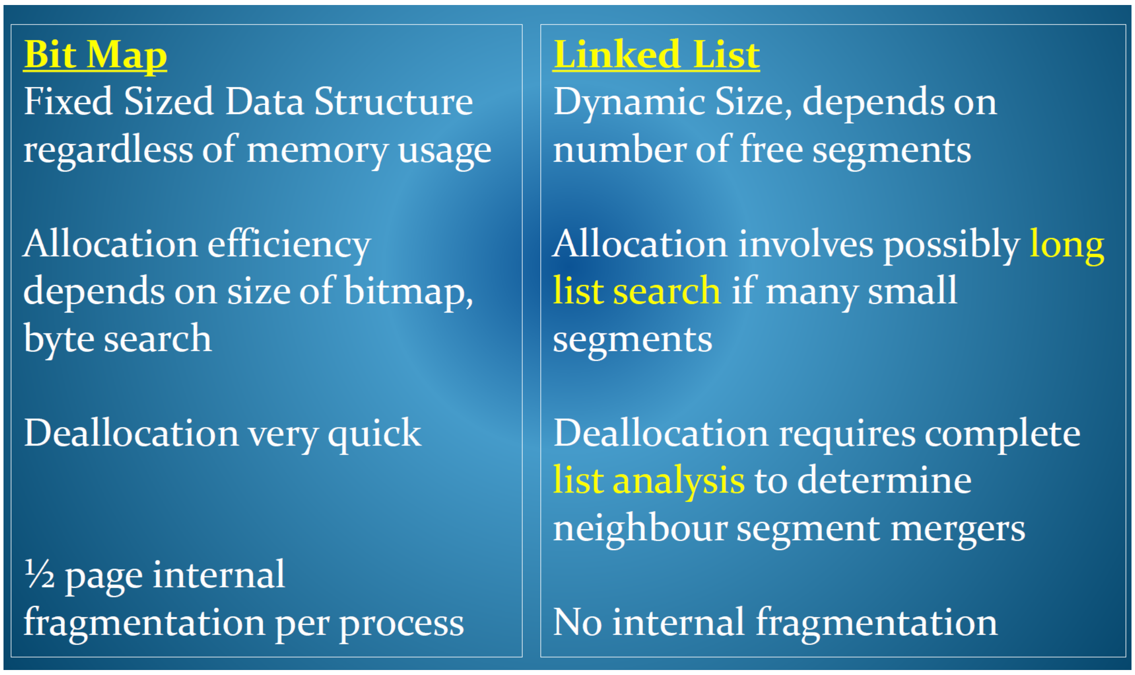
7. **Compare the bitmap approach to the linked list approach for keeping track of a free memory space.**

(1) 大小上：bitmap是固定大小的数据结构，而linked list是动态大小

(2)分配空间上：bitmap是根据大小确定分配的效率，linked list是需要一个很长list的搜索

(3)重新分配上：bitmap非常迅速，linked list需要对整个linked list进行操作

(4)碎片上：bitmap平均1/2的内部碎片，linked list不存在内部碎片



Bit Map	**Linked List**
Fixed Sized Data Structure regardless of memory usage	Dynamic Size, depends on number of free segments
Allocation efficiency depends on size of bitmap, byte search	Allocation involves possibly long list search if many small segments
Deallocation very quick	Deallocation requires complete list analysis to determine neighbour segment mergers
½ page internal fragmentation per process	No internal fragmentation

8. **Describe the operation of a paged memory architecture using diagrams to support your answer. Explain clearly the process of memory translation and the benefits of paged memory. Also state the disadvantages of the paged memory**

A memory address is divided into two parts, a **page number** and an **offset** within that page. The MMU exchanges the page number from the **logical address** generated by the processor, with the corresponding **physical page** found in the page table.

Advantages:

(1)It is not possible to **access the address space of other processes** because the page table won't contain pointers to pages for any other process.

(2)Do not all have to be **adjacent** to one another.

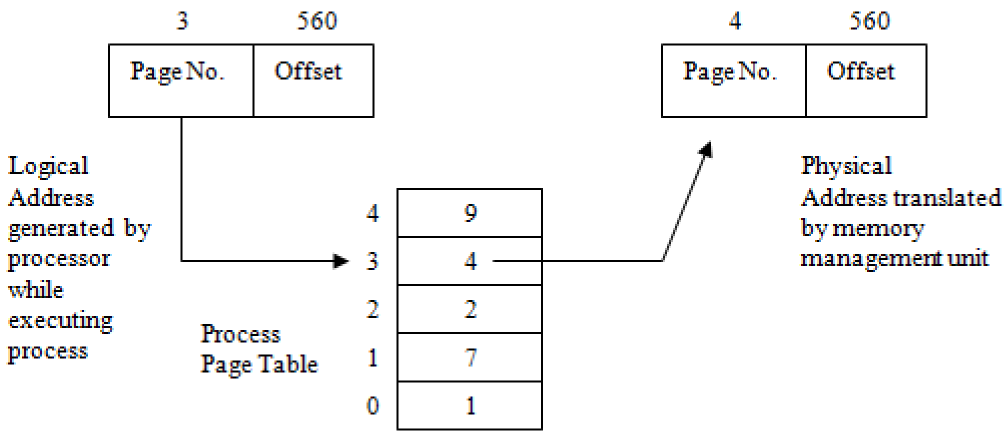
(3)It is easy to **relocate** or **expand** the address space of a process by altering the mapping of pages into its page table.

Disadvantages:

(1)Page tables can be **large** and need to be kept in **main memory** causing an **additional overhead**.

(2)On average approximately **½ a page per process would be wasted**. This phenomenon is known as internal fragmentation.

(3)Page mapping still **slows down the access time** to memory slightly.


3
560
Page No.
Offset
4
560
Page No.
Offset
Logical Address generated by processor while executing process
Process Page Table
4 9
3 4
2 2
1 7
0 1
Physical Address translated by memory management unit

9. **Describe the generic design requirements of a File System. List and explain some of the basic system calls typically provided for accessing and organising files.**

(1)The file system designer must decide how these blocks are **allocated** and **deallocated** in order to achieve the **maximum performance** from the file system as well as **ensuring file persistence and data integrity** . At the device driver level, the designer will have to be aware of the physical device characteristics.

The file system user is concerned with **how to access files , how to organise the internal structure of a file, organise collections of files into logical groupings and how to share them with other users**.

(2)**accessing files:**

Open Prepare a file to be referenced within the operating system

Close Prevent further reference until file is reopened

Create Build a new file within the collection

(3)**organising files:**

Read Copy some bytes from a file to a process

Write Output some bytes from a process to a file

Seek Search to a specified point in the file

开摆摆摆!