1.Describe the advantages of multiprogramming

提高CPU利用率，充分发挥并发性。程序间、设备间、设备与CPU间均并发工作。

有并发性、共享性、虚拟性、异步性。

**还可能出现的问题：批处理系统（batch system） ，多道批处理系统（multiprogramming batch system）分时操作系统（Time-sharing system）的技术特性**

**批处理：装入作业，自动从一个工作转向另一个，解决人机速度不匹配的问题，但CPU利用率仍然很低。**

**多道批处理：允许多个程序驻留内存，CPU利用率大大提高，但用户缺乏控制权，没有交互能力。**

**分时操作系统：将时间分为各个时间片，由于切换频率非常高使每个用户都觉得自己独占资源，允许多个用户同时交互使用计算机系统。**

2.What is the dual-mode of operation? What is the function of the mode bit?

为防止不正确的程序导致其他程序运行错误，将操作区分为用户态和内核态。当错误或中断发生时，硬件从用户态切换为内核态。双模式操作在错误用户程序发生时起到保护操作系统的作用。

用模式位来表示当前模式（用户态还是核心态）。

3.Describe the advantages of Two-level Multithreading Mode.

允许线程多对多也同时允许单对单绑定，有高效性的同时具有并发性，一个线程被阻塞后允许另一个继续执行，线程管理在用户空间进行。

4.Describe the differences between process and program.

程序是静态的是永久的，进程是程序在某个数据集上的执行，是暂时的、有周期的。

进程不能脱离程序而存在，程序规定了进程的动作。进程具有并发性，程序没有。

**还可能出现的问题：进程和线程的区别和联系**

**线程不能单独执行，一个进程至少包含一个主线程，进程是操作系统资源分配的基本单位，线程是CPU调度的基本单位。子进程和父进程有不同的代码和数据空间，而一个进程下的多个进程共享数据空间，每个线程有自己的堆栈和计数器。**

**用户级线程和内核级线程**

**用户级线程：只能在进程内，内核不知道用户级线程，如果用户级线程所在的进程有一个线程阻塞，那么整个进程都会被阻塞。效率较高。**

**内核级线程：效率比用户级线程低，内核级线程创建结束启用调度都在内核空间，但一个阻塞不会影响其他线程。在多处理机环境下，内核可以把内核级线程调度给不同的处理机。**

5.Describe the differences between Non-preemptive and preemptive.

非抢占式调度：一旦把处理机分配给某个进程后，只有进程自身终止或者进入挂起状态才会重新调度分配处理机。

抢占式调度：出现中断、IO操作完成、新的进程出现都可能导致系统根据某种调度原则去暂停某个正在执行的进程，将处理机重新分配给其他进程。

**还可能出现的问题：信号机制和中断机制的异同：**

**相同点：都采用了异步通信模式**

**都会暂停正在执行的进程，转而去处理相应服务程序，执行完后返回原来的断点继续执行。**

**都可以被屏蔽**

**不同点：信号处理程序在用户态中执行，中断处理程序在核心态运行。中断有优先级，信号没有优先级。中断响应是即时的，信号响应有延迟。**

6.Why is it important for the scheduler to distinguish IO-bound programs from Cpu-bound programs?

IO-bound 需要更多的IO时间较少的CPU时间，CPU-bound相反，而CPU与IO是可以并发工作的，所以在调度时混合选取这两种程序可以得到最高的CPU使用率，效率最大。

7.What is the main advantage of the micro kernel approach to operating system design?

更安全、更可靠、更容易延伸一个操作系统、更容易使操作系统适应新的体系结构。

**还可能出现的问题：分层方法（layed-structured）和模块化方法（module）的异同**

**共同点 都实现了系统的模块化设计，每一部分提供对外接口。**

**不同：模块化比分层方法更灵活，各个模块互相调用，而分层方法使用层层系统调用，开销很大。**

8.Describe the five state process model, describe what transitions are valid between the five states, and describe an event that might cause such a transition.

新建进程：经过操作系统准许进入 就绪状态

经过安排调度 进入运行状态

完全运行完毕进入 结束状态

时间片走完 运行变为就绪状态

因为处理IO 进入 等待状态

IO处理完毕 等待变为就绪状态

9.Describe the differences among short-term, medium-term, and long-term scheduling.

short-term 调度 选择哪一个进程下一个在CPU中运行 以毫秒为单位

Medium-term 将进程从存储中抹去并晚一点再次引入

Long-term 选择哪些进程需要被加载进入存储来运行 （几秒或者几分钟）

long-term 控制了有多少程序在主存中。

10.Describe the differences between shared memory and message passing in interprocess communication.

shared-memory 需要在地址空间创建一个共有空间来保证数据交互，且要保证多进程不能同时在一个地址写数据。

meassage-passing 需要建立交流连接，通过send/receive 操作进行数据交互，连接需要物理上的(通过shared memory 或者hardware bus) 以及是逻辑上的。

II. Give the probable screen output of the following program.

int main(){

pid\_t pid;

int a=8,b=5;

pid = fork(); //fork another process

if(pid<0){ //error occurred

fprintf(stderr,"Fork Failed");

exit(-1);

}

else if(pid>0){

printf("case l1 pid>0,a=%d,b=%d\n",a,b);

a=a\*2;

wait(NULL);

printf("casel2 pid>0,a=%d,b=%d\n",a,b);

}

else{

b=a-1;

printf("case 2 ,a=%d,b=%d\n",a,b);

exit(0);

} }

III. Consider the following set of processes, assumed to have arrived at time 0, in the order P1, P2, P3, P4, P5, with the length of the CPU-burst time given in milliseconds:

Process P1 P2 P3 P4 P5

Burst Time 9 3 2 1 5

Priority 3 1 5 4 2

a.Draw four Gantt charts that illustrate the execution of these processes using FCFS, non-preemptive SJF , non-preemptive priority(small number implies a higher priority) ,and RR(quantum = 1) scheduling .

b. What are the average turnaround time and the average waiting time of all process for each of the scehduling algorithms in part (a) resectively?

IV.Do you think the following algorithm can solute the critical section problem of P0 and P1? Give your reason .

Share variables int turn =0;

Share variables boolean flag[0]=flag[1]=false;

P0: do{

flag[0]=true;

while(turn!=0){

while(flag[1])

turn=0;

}

critical section

reminder;

}while(true);

P1: do{

flag[1]=true;

while(turn!=1){

while(flag[0])

turn=1;

}

critical section

reminder;

}while(true);