Operating System Homework

Zheng Chen
Stu. No. 2141601026
XJTUSE 42

December 2016

Contents

I Homework 1 2

1 Introduction 3

2 OS Structures 4

II Homework 2 5

3 Process 6

4 Thread 7

5 Scheduling 9

III Homework 3 13

6 Process Synchronization 14

7 Deadlock 17

IV Homework 4 18

8 Memory Management 19

9 Virtual Memory 21

V Homework 5 22

10 File System 23

11 I/O System 25

12 Secondary Storage 26

VI Appendix 27

Part I

Homework 1

Chapter 1

Introduction

1. What is the main advantage of multiprogramming?

   Increasing CPU utilization by dispatching works.

2. Define the essential properties of the following types of operating systems:

   (a) Batch

   import several jobs and continually run them.

   (b) Interactive

   support online user operation by I/O devices.

   (c) Time Sharing

   several users or processes share system resource by time.

   (d) Real time

   short latency, swift switch, high real-time availability.

   (e) Network

   processes commucate by network

   (f) Distributed

   abstract logical model, decouple from physical components

3. Consider the various definition of operating system. Consider whether the operating system should include applications such as web browsers and mail programs. Argue both that it should and that it should not, and support your answer.

   (a) Operating system **should** include general applications.

      i. It's hard to say an application that is absolutely a system application or a general application. (system applications such as *.NET Framework* in Windows) (definition)

      ii. It makes the operating system much easier to use. Nobody wants to view webpage by 'curl'. (user-friendly)

   (b) Operating system **should not** include general applications.

   General applications are not necessary for operating system. General applications help user working. But operating system should not assume which application users work with. User could choose applications themselves. (for a clean OS)

Chapter 2

OS Structures

1. What is the purpose of system calls?
   Providing the interactive interface between a running program and the operating system.
2. What is the main advantage of the layered approach to system design?
   (a) Serapating layer and layer is convenient to extend functions.
   (b) Decouple between layers.
   (c) Simply Requirements. (like a DAG)
3. What is the main advantage of the microkernel approach to system design?
   (a) Easy to extend.
   (b) More reliable.
   (c) Distributed calculations by Remote Procedure Call (sending requests).

Part II

Homework 2

Chapter 3

Process

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

   The main difference is the "distance" from the CPU to the process, which indicates the names of short-term, medium-term and long-term scheduling.

   (a) Speed
   Short-term ¿ Medium-term ¿ Long-term
   the less distance from the CPU to the process, the faster it is.

   (b) Frequency
   Short-term ¿ Medium-term ¿ Long-term
   the faster, the more frequent.

   (c) Place
   Short-term scheduling focus on inside CPU and main memory only (internal devices).
   Long-term scheduling focus on outside CPU and main memory (external devices).
   Medium-term scheduling works between internal devices and external devices.

   (d) Others
   Short-term scheduling only decides which process in the main memory will be processed by CPU.
   Long-term scheduling only decides which process in the external devices can enter main memory to be processed.
   Medium-term scheduling considers about if it is necessary to suspend a process to external memory to relief main memory.

2. Describe the actions taken by a kernel to context-switch between processes.

   Saving and loading context.

   (a) Saving process segments: text segment, data segment, stack segment and sharing memory.
   (b) Saving register data: PC, PSW, SP, PCBP, ISP, etc.
   (c) Saving virtual memory management data and interrupt stack data.

   and loading another process's corresponding data from somewhere.

Chapter 4

Thread

1. Discuss the difference between user-level thread and kernel-level thread.

   (a) Implementation
   Kernel-level thread is supported by kernel.
   User-level thread is supported by thread programming library.

   (b) Visibility
   Kernel knows kernel-level threads, but can notice user-level threads.
   Kernel takes user-level threads just as a process.
   So user-level thread blocking makes the whole process blocking, but kernel-level thread not.
   And user-level multi-thread can't gain more CPU time slice from kernel, but kernel-level thread can.

   (c) Privilege
   Operaing(creating, switching and destroying) kernel-level threads needs kernel privileges but it's free about user-level threads.

   (d) Maintainer
   Kernel-level thread's context is maintained by kernal.
   User-level thread's context is maintained by user process.

2. Which of the following components of program state are shared across threads in a multithreaded process?

   (a) Register values
   (b) Heap memory
   (c) Global variables
   (d) Stack memory

   b and c

   Heap memory belongs to the whole process runtime.

   Global variables belong to data segment.

   Each threads has its own register set and stack.

3. The program shown in Figure 4.11 uses the Pthreads API. What would be output from the program at LINE C and LINE P?

```
1  #include <pthread.h>
2  #include <stdio.h>
3  #include <sys/types.h>
4  #include <sys/wait.h>
5  #include <unistd.h>
6
7  int value = 0;
8  void *runner(void *param);
9
10 int main() {
11   int pid;
```

```
12    pthread_t tid;
13    pthread_attr_t attr;
14    pid = fork();
15    if (pid == 0) {
16      pthread_attr_init(&attr);
17      pthread_create(&tid, &attr, runner, NULL);
18      pthread_join(tid, NULL);
19      printf("CHILD: value=%d\n", value); /* LINE C */
20    } else if (pid > 0) {
21      wait(NULL);
22      printf("PARENT: value=%d\n", value); /* LINE P */
23    }
24  }
25
26  void *runner(void *param) {
27    value = 5;
28    pthread_exit(0);
29  }
```

Outputs:

```
1     CHILD: value=5
2     PARENT: value=0
```

Chapter 5

Scheduling

1. Why is it important for the scheduler to distinguish I/O-bound programs from CPU-bound programs?
   I/O-bound programs have more I/O bursts and less CPU bursts than CPU-bound programs.
   Scheduling Criteria:

   - Max CPU utilization
   - Max throughput and concurrency
   - Min turnaround time
   - Min waiting time
   - Min response time

   Scheduling dynamic process can improve and optimize these rules.

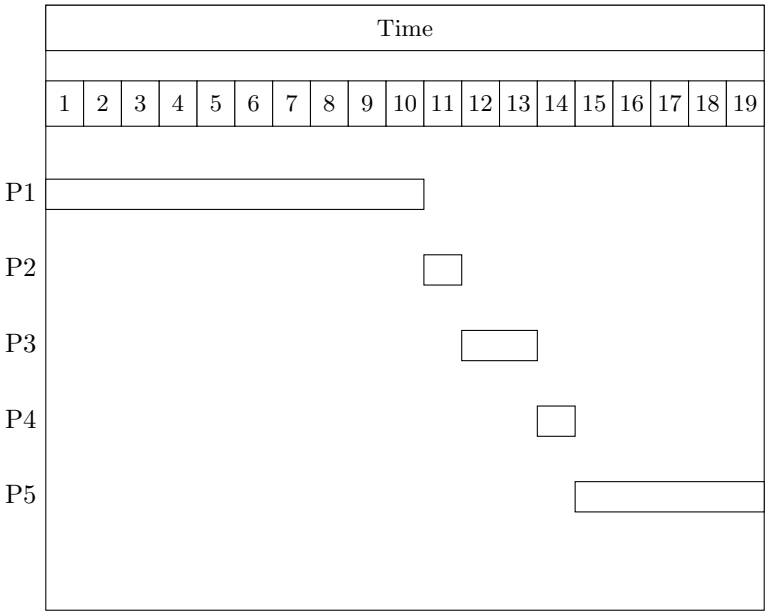
2. Consider the following set of processes, with the length of the CPU burst given in milliseconds

Process	Burst Time	Priority
P1	10	3
P2	1	1
P3	2	3
P4	1	4
P5	5	2

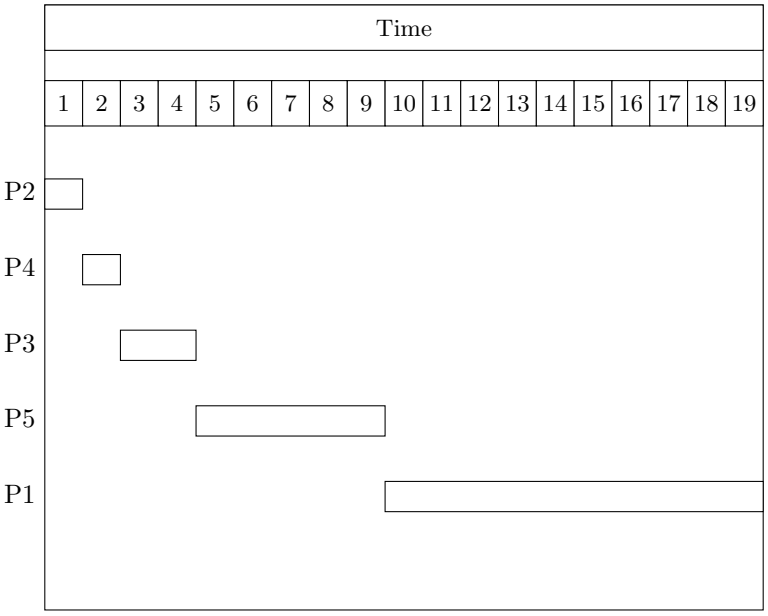
   The processes are assumed to have arrived in the order P1, P2, P3, P4, P5, all at time 0.

   (a) Draw four Gantt charts that illustrate the execution of these processes using the following scheduling algorithms: FCFS, SJF, nonpreemptive priority (a smaller priority number implies a higher priority), and RR (quantum = 1).

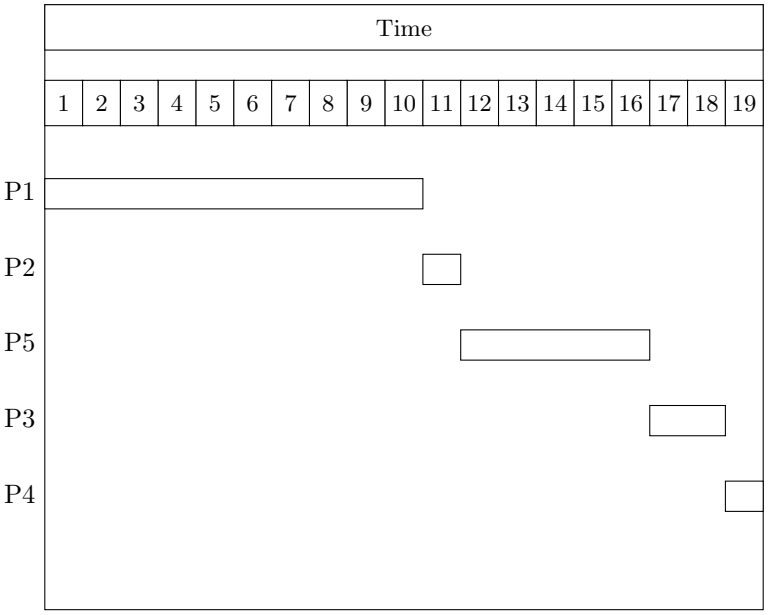
   - FCFS:



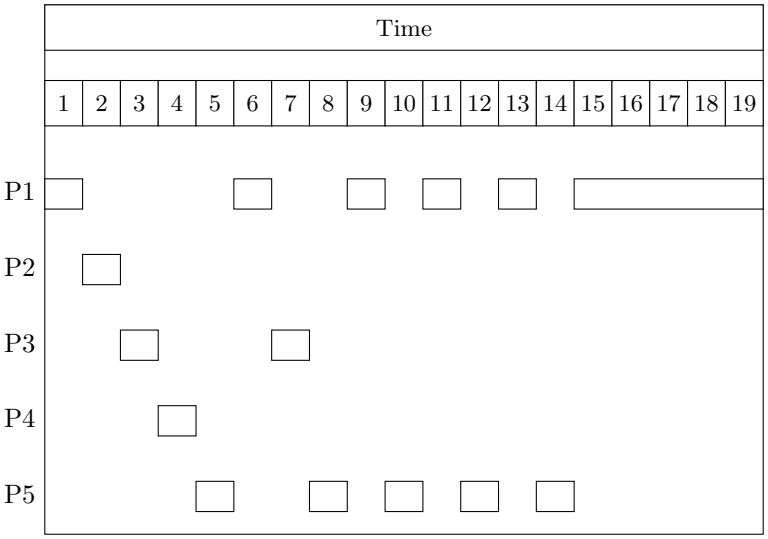
- SJF:



- nonpreemptive priority:



- RR (quantum = 1):



(b) What is the turnaround time of each process for each of the scheduling algorithms in part a?

- FCFS:

Process	Turnaround Time
P1	10
P2	11
P3	13
P4	14
P5	19

- SJF:

Process	Turnaround Time
P1	19
P2	1
P3	4
P4	2
P5	9

- nonpreemptive priority:

Process	Turnaround Time
P1	10
P2	11
P3	18
P4	19
P5	16

- RR:

Process	Turnaround Time
P1	19
P2	2
P3	7
P4	4
P5	14

(c) What is the waiting time of each process for each of the scheduling algorithms in part a?

- FCFS:

Process	Waiting Time
P1	0
P2	10
P3	11
P4	13
P5	14

- SJF:

Process	Waiting Time
P1	9
P2	0
P3	2
P4	1
P5	4

- nonpreemptive priority:

Process	Waiting Time
P1	0
P2	10
P3	16
P4	18
P5	11

- RR:

Process	Waiting Time
P1	9
P2	1
P3	5
P4	3
P5	9

(d) Which of the algorithms in part a results in the minimum average waiting time (over all processes)?

Algorithm	Average Waiting Time
FCFS	9.6
SJF(nonpreemptive)	9.4
SJF(preemptive)	3.2
nonpreemptive priority	11
RR	5.4

3. Which of the following scheduling algorithms could result in starvation?

   (a) First-come, first-served

   (b) Shortest job first

   (c) Round robin

   (d) Priority

   b and d.

4. The traditional UNIX scheduler enforces an inverse relationship between priority numbers and priorities: The higher the number, the lower the priority. The scheduler recalculates process priorities once per second using the following function:

   Priority = (Recent CPU usage / 2) + Base

   where base = 60 and recent CPU usage refers to a value indicating how often a process has used the CPU since priorities were last recalculated. Assume that recent CPU usage for process P1 is 40, process P2 is 18, and process P3 is 10. What will be the new priorities for these three processes when priorities are recalculated? Based on this information, does the traditional UNIX scheduler raise or lower the relative priority of a CPU-bound process?

   When priorities are recalculated:

Process	CPU usage	New Priority
P1	40	80
P2	18	69
P3	10	65

   the traditional UNIX scheduler does **not** raise or lower the relative priority of a CPU-bound process.

Part III

Homework 3

Chapter 6

Process Synchronization

6.4 Explain why spinlocks are not appropriate for single-processor systems yet are often used in multiprocessor systems.

Spinlock, which is a kind of busy-waiting lock, costs a single processor to check lock constantly. In single-processor systems, there's no more processor to do other tasks. Spinlock is a simple and quick method. In multiprocessor systems, there're idle processors can be used for spinlocks.

6.9 Show that, if the wait() and signal() semaphore operations are not executed atomically, then mutual exclusion may be violated.

Take wait() for example, wait() opeartions check the semaphore if its value is positive, and then decrease it. If wait() operations are not executed atomically, there may be many processes call wait() and check the positive semaphore together, and then decrease it without after checking, which cause too many processes went into critical section.

6.11 The Sleeping-Barber Problem. A barbershop consists of a waiting room with n chairs and abarber room with one barber chair. If there are no customers to be served, the barber goes to sleep. If a customer enters the barbershop and all chairs are occupied, then the customer leaves the shop. If the barber is busy but chairs are available, the customer sits in one of the free chairs. If the barber is asleep, the customer wakes up the barber. Write a program to coordinate the barber and the customers.

Listing 6.1: SleepingBarberProblem.java

```
1  import java.util.concurrent.Semaphore;
2
3  public class SleepingBarberProblem {
4    static final int chairCnt = 8;
5    static Semaphore chairs = new Semaphore(chairCnt);
6    static Semaphore barbers = new Semaphore(0);
7    static Semaphore sleepBarbers = new Semaphore(0);
8
9    static class Barber extends Thread {
10     String name;
11     public Barber(String aname) {
12       name = aname;
13       sleepBarbers.release();
14       System.out.println("[barber] new " + name);
15     }
16     public void run() {
17       try {
18         while (true) {
19           if (chairs.availablePermits() == chairCnt) {
20             if (barbers.tryAcquire()) {
21               System.out.println("[barber]" + name + " go to sleep");
22               sleepBarbers.release();
23             }
24           }
25         }
26       } catch (Exception e) {
```

```
27          e.printStackTrace();
28        }
29      }
30    }
31
32    static class Customer extends Thread {
33      String name;
34      public Customer(String aname) {
35        name = aname;
36        System.out.println("[customer] new " + name);
37      }
38      public void run() {
39        try {
40          // enter the barbarshop
41          if (chairs.tryAcquire()) {
42            System.out.println("[customer]" + name + " waiting for barber");
43            // try to wake up a barber
44            if (sleepBarbers.tryAcquire()) {
45              System.out.println("[customer]" + name + " woke up a barber");
46              barbers.release();
47            }
48            barbers.acquire();
49            System.out.println("[service]" + name + " begin");
50            Thread.sleep(500); // for a while
51            System.out.println("[service]" + name + " end");
52            barbers.release();
53            chairs.release();
54            System.out.println("[customer] Customer " + name + " exited");
55          } else {
56            // all chairs are occupied
57            System.out.println("[warning]" + name +
58                               " can not find chairs and leaved");
59          }
60        } catch (Exception e) {
61          e.printStackTrace();
62        }
63      }
64    }
65
66    public static void main(String[] args) {
67      new Barber("zccz14").start();
68      try {
69        String prefix = "wuke";
70        int cnt = 0;
71        while (true) {
72          for (int i = 0; i < 10; i++) {
73            new Customer(prefix + (++cnt)).start();
74          }
75          Thread.sleep(6000);
76        }
77      } catch (Exception e) {
78        e.printStackTrace();
79      }
80    }
81  }
```

1. There is a kind of definition of P, V operation:

```
P(s):
  s := s-1;
  if s < 0 then
```

```
    push the process into the tail of the corresponding waiting queue;
V(s):
  s := s + 1;
  if s <= 0 then
    pop a process P from the tail of the corresponding waiting queue;
    push P into ready queue;
```

(a) Is there a problem with the above definition of P, V operation?

(b) Implement a mutual exclusion mechanism, under which N process to compete for the use of a shared variable, with the above P, V operation.

2. The Second Reader-Writer Problem: Writer First

   - Many reader can read in the same time.
   - Writers exclude any others.
   - Writers have higher priority than readers. Readers should wait if there is any writers.

   Regard N students and a teacher as processes. People should enter or leave the exam room one by one follow the principle of first come. The teacher should dispatch papers after all the N students entered the room. Student should leave the room after submitted their paper. The teacher should wait for all paper submitted and then leave the room.

   (a) How many processes do we need to set?

   (b) Try to use P, V operation to describe synchronization and mutual exclusion in the problem.

Chapter 7

Deadlock

7.1 Consider the traffic deadlock depicted in Figure 7.9.

(a) Show that the four necessary conditons for deadlock indeed hold in this example.

(b) State a simple rule for avoiding deadlocks in this system.

7.11 Consider the following snapshot of a system:

	Allocation	Max	Available
P0	0012	0012	1520
P1	1000	1750	
P2	1354	2356	
P3	0632	0652	
P4	0014	0656	

Answer the following questions using the banker's algorithm:

(a) What is the content of the matrix Need?

(b) Is the system in a safe state?

(c) If a request from process P1 arrives for (0,4,2,0), can the request be granted immediately?

Part IV

Homework 4

Chapter 8

Memory Management

8.1 Explain the difference between internal and external fragmentation.

They are both waste of memory. Internal fragmentation is from allocated aligned memory blocks. Allocating aligned memory blocks is convenient for OS, while it brings internal fragmentation. External fragmentation is from the arrangement problem. OS can use the whole memory to satisfy a memory allocating request. External fragmentation will accumulate while memory allocating and free. OS could rearrange programs to reduce external fragmentation, but there is no way to reduce internal fragmentation.

8.3 Given five memory partitions of 100KB, 500KB, 200KB, 300KB, and 600KB (in order), how would each of the first-fit, best-fit, and worst-fit algorithms place processes of 212KB, 417KB, 112KB, and 426KB (in order)? Which algorithm makes the most efficient use of memory?

first-fit

memory partition (KB)					Note
100	500	200	300	600	initial state
100	288	200	300	600	allocated 212KB
100	288	200	300	183	allocated 417KB
100	176	200	300	183	allocated 112KB
100	176	200	300	183	allocating 426KB, failed

best-fit

memory partition (KB)					Note
100	500	200	300	600	initial state
100	500	200	88	600	allocated 212KB
100	83	200	88	600	allocated 417KB
100	83	88	88	600	allocated 112KB
100	83	88	88	174	allocated 426KB

worst-fit

memory partition (KB)					Note
100	500	200	300	600	initial state
100	500	200	300	388	allocated 212KB
100	83	200	300	388	allocated 417KB
100	83	200	300	276	allocated 112KB
100	83	200	300	276	allocating 426KB, failed

Obviously the **best-fit** algorithm makes the most efficient use of memory.

8.9 Consider a paging system with the page table stored in memory.

(a) If a memory reference takes 200 nanoseconds, how long does a paged memory reference take?
In this scheme every data/instruction access requires two memory accesses.

$$T = 2 \times 200\text{ns} = 400\text{ns}$$

(b) If we add TLBs, and 75 percent of all page-table reference are found in the TLBs, what is the effective memory reference time? (Assume that finding a page-table entry in the TLBs takes zero time, if the entry is there)

$$T = 200\text{ns} \times (1 \times 75\% + 2 \times 25\%) = 250\text{ns}$$

8.12 Consider the following segment table:

Segment	Base	Length
0	219	600
1	2300	14
2	90	100
3	1327	580
4	1952	96

What are the physical addresses for the following logical addresses?

$$A_p = Base + A_l, (A_l < Length)$$

(a) 0, 430

$$A_p = 219 + 430 = 649$$

(b) 1, 10

$$A_p = 2300 + 10 = 2310$$

(c) 2, 500

$$A_p = 90 + 500 = 590, (\text{Illegal})$$

(d) 3, 400

$$A_p = 1327 + 400 = 1727$$

(e) 4, 112

$$A_p = 1952 + 96 = 2048$$

Chapter 9

Virtual Memory

9.4 A certain computer provides its users with a virtual-memory space of 2^{32} bytes. The computer has 2^{18} bytes of physical memory. The virtual memory is implemented by paging, and the page size is 4,096 bytes. A user process generates the virtual address 11123456. Explain how the system establishes the corresponding physical location. Distinguish between software and hardware operations.

9.10 Consider a demand-paging system with the following time-measured utilization:

CPU utilization	20%
Paging disk	97.7%
Other I/O devices	5%

For each of the following, say whether it will (or is likely to) improve CPU utilization. Explain your answer.

(a) Install a faster CPU.

(b) Install a bigger paging disk.

(c) Increase the degree of multiprogramming.

(d) Decrease the degree of multiprogramming.

(e) Install more main memory.

(f) Install a faster hard dist or multiple controllers with multiple hard disks.

(g) Add prepaging to the page-fetch algorithms.

(h) Increase the page size.

9.13 A page-replacement algorithm should minimize the number of page faults. We can achieve this minimization by distributing heavily used pages evenly over all of memory, rather than having them compete for a small number of page frames. We can associate with each page frame a counter of the number of pages associated with that frame. Then, to replace a page, we can search for the page frame with the smallest counter.

(a) Define a page-replacement algorithm using this basic idea. Specifically address these problems:

i. What the initial value of the counters is?

ii. When counters are increased?

iii. When counters are decreased?

iv. How the page to be replaced is selected?

(b) How many page faults occur for your algorithm for the following reference string, with four page frames?

1, 2, 3, 4, 5, 3, 4, 1, 6, 7, 8, 7, 8, 9, 7, 8, 9, 5, 4, 5, 4, 2

(c) What is the minimum number of page faults for an optimal page-replacement strategy for the reference string in part b with four frames?

Part V

Homework 5

Chapter 10

File System

10.1 Consider a file system where a file can be deleted and its disk space reclaimed while links to that file still exist. What problems may occur if a new file is created in the same storage area or with the same absolute path name? How can these problems be avoided?

There is an unexpected way to access the new file through the links to the old file.

Take microsoft windows for example, most of links are logical links called *shortcut*, which links to a logical file name. Redirect technique was used for logical-level file linking. When access the links linking to deleted files, we would be redirected to another logical filename and finally get an error——*"the target was moved or deleted"*. If the new file was stored with the same absolute path name, we can access the new file by the undeleted *shortcut*.

To ensure that all the links are deleted before the corresponding disk space being reclaimed is the **only** way to avoid the misleading link problem [1]thoroughly.

11.1 Consider a file system that uses a modified contiguous-allocation scheme with support for extents. A file is a collection of extents, with each extent corresponding to a contiguous set of blocks. A key issue in such system is the degree of variability in the size of the extents. What are the advantages and disadvantages of the following schemes?

(a) All extents are of the same size, and the size is predetermined.
Lowest complexity and lowest flexibility.

(b) Extents can be of any size and are allocated dynamically.
Highest complexity and highest flexibility.

(c) Extents can be of a few fixed sizes, and these sizes are predetermined.
Intermediate complexity and intermediate flexibility.

Higher flexibility indicates higher complexity with more difficult implementation and higher preprocess cost.

11.2 What are the advantages of the variant of linked allocation that uses a FAT to chain together the blocks of a file?

FAT is much fewer than the whole disk, so FAT can be well cached in memory. When we access the middle of a file record, we can lookup the FAT and find the related block number, and then directly access the corresponding block in the disk. In contrast to traversing block in the disk, using FAT saved a lot of disk-access time.

11.3 Consider a system where free space is kept in a free-space list.

(a) Consider a file system similar to the one used by UNIX with indexed allocation. How many disk I/O operations might be required to read the contents of a small local file at /a/b/c? Assume that none of the disk blocks is currently being cached.
Reading the contents of the small local file at /a/b/c need 4 disk I/O operations.

i. Reading / (the root directory)
ii. Reading /a

[1]But we hope the links still exist and automatically link to the new file in some cases. To avoid the problem by using a good naming style is more effective.

iii. Reading /a/b

iv. Reading /a/b/c

(b) Suggest a scheme to ensure that the pointer to the free space list is never lost as a result of memory failure.

Store the free-space list pointer in a stable storage, such as a disk, perhaps in RAID [2].

[2] Why not try using offsite disaster recovery technology? :)

Chapter 11

I/O System

Chapter 12

Secondary Storage

12.2 Suppose that a disk drive has 5,000 cylinders, numbered 0 to 4999. The drive is currently serving a request at cylinder 143, and the previous request was at cylinder 125. The queue of pending requests, in FIFO order, is:

86, 1470, 913, 1774, 948, 1509, 1022, 1750, 130

Starting from the current head position, what is the total distance (in cylinders) that the disk arm moves to satisfy all the pending requests for each of the following disk-scheduling algorithms?

- FCFS

$$\begin{aligned}\text{Total Distance} =& |143-86| + |86-1470| + |1470-913| \\ &+ |913-1774| + |1774-948| + |948-1509| \\ &+ |1509-1022| + |1022-1750| + |1750-130| \\ =& 7081(\text{Cylinders})\end{aligned}$$

- SSTF

$$\text{Total Distance} = |143-86| + |1774-86| = 1745(\text{Cylinders})$$

- SCAN

$$\text{Total Distance} = |143-4999| + |4999-86| = 9769(\text{Cylinders})$$

- LOOK

$$\text{Total Distance} = |143-1774| + |1774-86| = 3319(\text{Cylinders})$$

- C-SCAN

$$\text{Total Distance} = |143-4999| + |4999-0| + |0-130| = 9985(\text{Cylinders})$$

- C-LOOK

$$\text{Total Distance} = |143-1774| + |1774-86| + |86-130| = 3363(\text{Cylinders})$$

Part VI

Appendix