COS 222 - Semester Test 2 Notes

regan

September 22, 2016

Contents

1 Chapter 7 - Memory Management

1.1 Memory Management Requirements

A Frame is a fixed-length block of main memory

A Page is a fixed-length block of data that resides in secondary memory (such as a disk). A page may temporarily be copied into a frame of main memory.

A Segment is a variable-length block of data that resides in secondary memory. An entire segment may be temporary copied into an available region of main memory (this process is known as segmentation) or the segment may be divided into pages which can individually be copied into main memory (combined segmentation and paging).

1.1.1 Relocation

We need our system to be able to move the location where a process.

1.1.2 Protection

Each process should protected against unwanted interference by other processes, whether accidental or intentional. Thus programs in other processes should not be able to reference memory loctions in a process for reading or writing without sufficient permission.

1.1.3 Sharing

In order to minimize redundancy, we would ideally like processes to be able to share blocks of memory, otherwise there will be numerous instances where this data would have to be duplicated.

1.1.4 Logical Organization

The main memory of a computer system is most likely to be arranged as a linear, or one-dimensional, address space of either bytes or words.

1.1.5 Physical Organization

Memory is organised in at least two levels, namely primary and secondary memory

The main memory available for a program plus its data may be insufficient. In that case, the programmer must engage in a practise known as overlaying.

1.2 Memory Partitioning

1.2.1 Fixed Partitioning

This is the idea that memory is split into certain sizes

1. Partition Sizes

   Main memory utilization is extremely inefficient. Any program, no matter how small, occupies an entire partition. A program will be forced to occupy some partition that is larger than its maximum size. This is wasted space, and we call this waste *internal f

2. Placement Algorithm

1.2.2 Dynamic Partitioning

With dynamic partitioning, the partitions are of variable length and number.

1. The Buddy System

1.3 Paging

Both unequal fixed-size and variable-size partitions are inefficient in the use of memory; the former results in **internal fragmentation** and the latter in **external fragmentation**.

Suppose, however, that memory is partitioned into equal fixed-size chunks that are relatively small, and that each process is also divided into small fixed-sized chunks of the same size. The chunks of the process are known as **pages**, which can be assigned into chunks of memory, **frames**.

1.4 Segmentation

A user program can be subdivided using segmentation, in which the program and its associated data are divided into a number of **segments**. It is not required that all segments of all programs be of the same length, although there is a maximum segment length.

The difference between segmentation and paging is that

Paging	Segmentation
Transparent to programmer	Involves the programmer
No seperate protection	
No seperate compiling	
No shared code.	

2 Chapter 8 - Virtual Memory

2.1 Hardware Control Structures

2.1.1 Locality and Virtual Memory

2.1.2 Paging

2.1.3 Segmentation

2.1.4 Combined Paging and Segmentation

2.1.5 Protection and Sharing

2.2 Operating System Software

2.2.1 Fetch Policy

1. Demand Paging

   This is the more simple of the two fetching policies. Pages are brought into memory when they are requested, which effectively means that a page fault occurs. This means that at the beginning of the systems' run-time, page faults will be numerous, but will decrease as the popular pages get proceedurally added to main memory.

2. Prepaging

   This policy attempts to predict the realistic future page use, usually by means of the **Principle of Locality**. Rather than simply retrieving one page, it retrieves a certain amount of its neighbours.

2.2.2 Placement Policy

2.2.3 Replacement Policy

When the memory we have available to load pages becomes full, we need certain heuristics that allow us to logically replace and evict certain pages.

1. Basic Algorithms

   (a) Optimal

   The optimal replacement policy is a theoretical concept that could only be implemented with perfect information about the past, present and future of the system.

(b) Least Recently Used (LRU)

In this replacement policy (which happens to be one of the most popular).

(c) First-in-First-Out (FIFO)

This replacement policy will pereferencially remove older pages to newer ones.

(d) Clock

2. Page Buffering

An interesting strategy that can improve paging performance and allow the use of a simpler paging replacement policy is that of page buffering.

2.2.4 Resident Set Management

2.2.5 Cleaning Policy

These are the policies used to decide which pages should be removed from main memory. These poilicies mirror/complement the fetching policies. Cleaning policies are important

1. Demand

With demand cleaning, a page is written out to secondary memory only when it has been selected for replacement.

2. Precleaning

The precleaning policy will write to seconday memory early such that pages can be expelled from main memory in batches.

2.2.6 Load Control

2.3 Unix and Solaris Memory Management

2.4 Linux Memory Management

2.5 Windows Memory Management

2.6 Android Memory Management

3 Chpater 9 - Uniprocessor Scheduling

3.1 Types of Scheduling

3.1.1 Long-Term Scheduling

The long-term scheduler determines which programs are admitted.

3.1.2 Medium-Term Scheduling

Medium-term sheculing is part of the swapping function. This scheduling is the decision of what processes should be partiallly or fully in main memory.

3.1.3 Short-Term Scheduling

Also known as the **dispatcher**, short-term scheduling involves deciding what process should be executed next by the processor.