Segmentation in Operating System with Memory Management

M.Priyadharshini, A.R.Vithy and E.M.Sowmiya,

1,2,3Department of information and technology, Sri Krishna Arts and Science College, Kuniamuthur, Coimbatore, India

Abstract-- This paper illustrates about the memory management in operating system and it will demonstrate the basic types of operating system with its segmentation architecture and its allocation.

Keywords-- Memory, processor resources, allocation, multitasking, software.

I. INTRODUCTION

A. Operating System

An operating system (OS) is system software that manages computer hardware and software resources and provides common services for computer programs. All computer programs, excluding firmware, require an operating system to function.

Time-sharing operating systems schedule tasks for efficient use of the system and may also include accounting software for cost allocation of processor time, mass storage, printing, and other resources.

For hardware functions such as input and output and memory allocation, the operating system acts as an intermediary between programs and the computer hardware, although the application code is usually executed directly by the hardware and frequently makes system calls to an OS function or is interrupted by it. Operating systems are found on many devices that contain a computer—from cellular phones and video game consoles to web servers and supercomputers.

The dominant desktop operating system is Microsoft Windows with a market share of around 82%. OS X by Apple Inc. is in second place (9.8%), and Linux is in third position (1.5%). In the mobile (smartphone and tablet combined) sector Android by Google is dominant with 65% and iOS by Apple is placed second with around 25%. Linux is dominant in the server and supercomputing sectors. Other specialized classes of operating systems, such as embedded and real-time systems, exist for many applications.

B. Types of Operating System

a. Single- and multi-tasking

A single-tasking system can only run one program at a time, while a multi-tasking operating system allows more than one program to be running in concurrency. This is achieved by time-sharing, dividing the available processor time between multiple processes that are each interrupted repeatedly in time slices by a task-scheduling subsystem of the operating system. Multi-tasking may be characterized in preemptive and cooperative types. In preemptive multitasking, the operating system slices the CPU time and dedicates a slot to each of the programs. Unix-like operating systems, e.g., Solaris, Linux, as well as AmigaOS support preemptive multitasking. Cooperative multitasking is achieved by relying on each process to provide time to the other processes in a defined manner. 16-bit versions of Microsoft Windows used cooperative multi-tasking. 32-bit versions of both Windows NT and Win9x, used preemptive multi-tasking.

b. Single- and multi-user

Single-user operating systems have no facilities to distinguish users, but may allow multiple programs to run in tandem. A multi-user operating system extends the basic concept of multi-tasking with facilities that identify processes and resources, such as disk space, belonging to multiple users, and the system permits multiple users to interact with the system at the same time. Time-sharing operating systems schedule tasks for efficient use of the system and may also include accounting software for cost allocation of processor time, mass storage, printing, and other resources to multiple users.

c. Distributed

A distributed operating system manages a group of distinct computers and makes them appear to be a single computer. The development of networked computers that could be linked and communicate with each other gave rise to distributed computing. Distributed computations are carried out on more than one machine. When computers in a group work in cooperation, they form a distributed system.

d. Templated

In an OS, distributed and cloud computing context, templating refers to creating a single virtual machine image as a guest operating system, then saving it as a tool for multiple running virtual machines. The technique is used both in virtualization and cloud computing management, and is common in large server warehouses.

e. Embedded

Embedded operating systems are designed to be used in embedded computer systems. They are designed to operate on small machines like PDAs with less autonomy. They are able to operate with a limited number of resources. They are very compact and extremely efficient by design. Windows CE and Minix 3 are some examples of embedded operating systems.

f. Real-time

A real-time operating system is an operating system that guarantees to process events or data by a specific moment in time. A real-time operating system may be single- or multi-tasking, but when multitasking, it uses specialized scheduling algorithms so that a deterministic nature of behavior is achieved. An event-driven system switches between tasks based on their priorities or external events while time-sharing operating systems switch tasks based on clock interrupts.

g. Library

A library operating system is one in which the services that a typical operating system provides, such as networking, are provided in the form of libraries. These libraries are composed with the application and configuration code to construct...
unikernels – which are specialized, single address space, machine images that can be deployed to cloud or embedded environments.

C. Segmentation
Memory-management scheme that supports user view of memory.

A program is a collection of segments. A segment is a logical unit such as:
1. main program
2. procedure
3. function
4. method
5. object
6. local variables
7. global variables
8. common block
9. stack
10. symbol table
11. arrays

D. Segmentation Architecture
Logical address consists of a two tuple: <segment-number, offset>.

Segment table – maps two-dimensional physical addresses; each table entry has:
1. Base – contains the starting physical address where the segments reside in memory.
2. Limit – specifies the length of the segment.

Segment-table base register (STBR) points to the segment table’s location in memory.

Segment-table length register (STLR) indicates number of segments used by a program; segment number s is legal if s < STLR.

a. Relocation.
1. Dynamic
2. by segment table
4. shared segments
5. same segment number
6. Allocation.
7. first fit/best fit
8. external fragmentation

Protection. With each entry in segment table associate:
1. validation bit = 0 _ illegal segment
2. read/write/execute privileges

E. Memory Segmentation
Memory segmentation is the division of a computer’s primary memory into segments or sections. In a computer system using segmentation, a reference to a memory location includes a value that identifies a segment and an offset (memory location) within that segment. Segments or sections are also used in object files of compiled programs when they are linked together into a program image and when the image is loaded into memory.

Segments usually correspond to natural divisions of a program such as individual routines or data tables so segmentation is generally more visible to the programmer than paging alone. Different segments may be created for different program modules, or for different classes of memory usage such as code and data segments. Certain segments may be shared between programs.

F. Hardware Implementation
In a system using segmentation, computer memory addresses consist of a segment id and an offset within the segment. A hardware memory management unit (MMU) is responsible for translating the segment and offset into a physical memory address, and for performing checks to make sure the translation can be done and that the reference to that segment and offset is permitted.

Each segment has a length and set of permissions (for example, read, write, execute) associated with it. A process is only allowed to make a reference into a segment if the type of reference is allowed by the permissions, and if the offset within the segment is within the range specified by the length of the segment. Otherwise, a hardware exception such as a segmentation fault is raised.

Segments may also be used to implement virtual memory. In this case each segment has an associated flag indicating whether it is present in main memory or not. If a segment is accessed that is not present in main memory, an exception is raised, and the operating system will read the segment into memory from secondary storage.

Segmentation is one method of implementing memory protection. Paging is another, and they can be combined. The size of a memory segment is generally not fixed and may be as small as a single byte.
Segmentation has been implemented in several different ways on different hardware, with or without paging. Intel x86 memory segmentation does not fit either model and is discussed separately below, and also in greater detail in a separate article.

Segmentation without paging:

Associated with each segment is information that indicates where the segment is located in memory— the segment base. When a program references a memory location the offset is added to the segment base to generate a physical memory address.

An implementation of virtual memory on a system using segmentation without paging requires that entire segments be swapped back and forth between main memory and secondary storage. When a segment is swapped in, the operating system has to allocate enough contiguous free memory to hold the entire segment. Often memory fragmentation results in there being not enough contiguous memory even though there may be enough in total.

Segmentation with paging:

Instead of an actual memory location the segment information includes the address of a page table for the segment. When a program references a memory location the offset is translated to a memory address using the page table. A segment can be extended simply by allocating another memory page and adding it to the segment's page table.

An implementation of virtual memory on a system using segmentation with paging usually only moves individual pages back and forth between main memory and secondary storage, similar to a paged non-segmented system. Pages of the segment can be located anywhere in main memory and need not be contiguous. This usually results in a reduced amount of input/output between primary and secondary storage and reduced memory fragmentation

**CONCLUSION**

Hence with this we have understood about the indispensable concept of operating system and its types. We also seen about the architecture of segmentation in memory and also a short outlook of memory management.

**References**

[1] [http://whatis.techtarget.com/definition/segmented-memory](http://whatis.techtarget.com/definition/segmented-memory)

[2] [https://www.tutorialspoint.com/operating_system/os_memory_management.htm](https://www.tutorialspoint.com/operating_system/os_memory_management.htm)

[3] [https://en.wikipedia.org/wiki/Memory_segmentation](https://en.wikipedia.org/wiki/Memory_segmentation)

[4] [http://www.gitam.edu/eresource/comp/gvr(os)/8.5.html](http://www.gitam.edu/eresource/comp/gvr(os)/8.5.html)