Early Paradigms of Memory Management

Lecture #2

Department of Computer Science and Technology
University of Bedfordshire

Written by David Goodwin,
based on the lecture series of Dayou Li
and the book *Understanding Operating Systems 4thed.*
by *I.M.Flynn and A.McIver McHoes* (2006).

Operating Systems, 2013

Outline

Introduction

Single-user

Fixed partitions

Dynamic partitions

Relocatable partitions

PROBLEM-SOLVING

- We “stand on the shoulder of giants”
- A problem has to have a solution
- No solution is perfect
- Solutions are being refined for various reasons
- We are in the process of refining solutions

SINGLE-USER CONTIGUOUS SCHEME

- Each user is given access to all available main memory for each job and jobs are processed sequentially, one after another

Algorithm 1 Algorithm for loading a program

```
Store first memory location of programs into base register
Set program counter
Read first instruction of program
Increase program counter by number of bytes in instruction
if Test – has the last instruction been reached?  then
    If yes, then stop loading program
else
    If no, the continue with Step 6
end if
if Test – is program counter greater than memory size? then
    If yes, then stop loading program
else
    If no, then continue with Step 7
end if
Load instruction in memory
Read next instruction of program
Goto step 4
```

SINGLE-USER CONTIGUOUS SCHEME

- Analysis
  - Advantages:
    - Logic is simple
    - Implementation is straightforward
    - Only 2 hardware items are required – register as the base register and accumulator as the program counter
  - Disadvantage:
    - If program size is larger than memory size, then the program cannot run
    - It does not support multiprogramming as it can only handle one job at a time
    - The entire program must be contiguously stored in memory

FIXED PARTITIONS

- Main memory is partitioned into a fix number of partitions the sizes of which are also fixed
- Each of multiple users can access to a partition
- The configurations of the partitions cannot be changed when the computer system is operating
- The partitions can only be reconfigured when the computer system is shut down
- Fixed partitions is the first attempt to allow multiprogramming

FIXED PARTITIONS

Algorithm 2 Algorithm of loading jobs to memory

```
Determine job's requested memory size
if job_size > size of the largest partition then
    Reject the job
else
    Continue with Step 3
end if
Set counter to 1
while counter <= number of partitions in memory do
    if job_size > memory_partition_size(counter) then
        Counter = counter + 1
    else
        if memory_partition(counter) = "free" then
            Load job to memory_partition (counter)
            Change memory_partition_status(counter) to "busy"
            Goto Step 1 to handle the next job
        else
            counter = counter + 1
        end if
    end if
end while
No partition is available at this time, put job in waiting queue
Goto Step 1 to handle the next job in line
```

FIXED PARTITIONS

Example

FIXED PARTITIONS

Introduction

Single-user

Fixed partitions

Dynamic partitions

Relocatable partitions

- Analysis
  - Advantage:
    - Allowing multiprogramming/multi-user
  - Problem introduced
    - Protection of job's memory space – once a partition is allocated for a job, no other job could be allowed to use this partition
    - A program still must be entirely and contiguously stored in a partition
    - Space in some partitions have been wasted
    - Some jobs cannot be loaded

DYNAMIC PARTITIONS

- Size and number of partitions are determined according to jobs which are initially in the job list and their sizes
  - Advantage:
    - Jobs are given as much memory as they request when they are loaded for processing
  - Problem:
    - When new jobs arrive, the problem of wasting memory space takes place again
    - Some new jobs cannot be loaded

DYNAMIC PARTITIONS

- Example
  - Determine partitions according to jobs in the job list
  - Assign jobs to the main memory

partition size	memory address	access	partition status
10k	200k	job 1	busy
15k	210k	job 2	busy
20k	225k	job 3	busy
50k	245k	job 4	busy

DYNAMIC PARTITIONS

- When job 1 and job 4 end

partition size	memory address	access	partition status
10k	200k		free
15k	210k	job 2	busy
20k	225k	job 3	busy
50k	245k		free

- When new jobs 5 (5k) and 6 (30k) arrive

partition size	memory address	access	partition status
5k	200k	job 5	busy
5k	205k		free
15k	210k	job 2	busy
20k	225k	job 3	busy
30k	245k	job 6	busy
20k	275k		free

- When job 3 ends

partition size	memory address	access	partition status
5k	200k	job 5	busy
5k	205k		free
15k	210k	job 2	busy
20k	225k		free
30k	245k	job 6	busy
20k	275k		free

DYNAMIC PARTITIONS

- When jobs 7(10k) and 8(30k) arrive (job 8 cannot be loaded)

partition size	memory address	access	partition status
5k	200k	job 5	busy
5k	205k		free
15k	210k	job 2	busy
10k	225k	job 7	busy
10k	235k		free
30k	245k	job 6	busy
20k	275k		free

DYNAMIC PARTITIONS

Algorithm 3 Best-fit (the smallest partition fitting the requirement)

```
Initialise memory_block(0) = 99999
Computer initial_memory_waste = memory_block(0) − job_size
Inialise subscript = 0
Set counter to 1
while counter <= number of blocks in memory do
    if job_size > memory_size(counter) then
        counter = counter + 1
    else
        memory_waste = memory_size(counter) − job_size
        if initial_memory_waste > memory_waste then
            subscript = counter
            Initial_memory_waste = memory_wast
            counter = counter + 1
        end if
    end if
end while
if subscript = 0 then
    put job in waiting list
else
    load job into memory(subscript)
    adjust free/busy memory lists
end if
Fetch next job
```

FIXED PARTITIONS

Algorithm 4 First-fit (first partition fitting the requirement)

```
Set counter to 1
while counter $<=$ number of blocks in memory do
    if job_size $>$ memory_size(counter) then
        counter $=$ counter $+ 1$
    else
        load job into memory(counter)
        adjust free/busy memory lists
        go to step 4
    end if
end while
Put job in waiting queue
Fetch next job
```

RELOCATABLE DYNAMIC PARTITIONS

- Deallocation
  - Deallocation is about releasing memory blocks
  - Two tasks:
    - Set partition status to free when a job ends
    - Combine free blocks whenever possible

RELOCATABLE DYNAMIC PARTITIONS

Algorithm 5 Algorithm to delocate memory blocks

```
if job_locaton is adjacent to one or more free blocks then
    if job_location is between two free blocks then
        merge the three blocks
        mem_size(counter−1)=mem_size(counter−1)+job_size+mem_size(counter+1)
        mem_status(counter+1)=null entry
    else
        merge both blocks into one
        mem_size(counter−1)=mem_size(counter−1)+job_size
    end if
else
    search for null entry in free memory list
    enter job_size and beginning address in the entry slot
    set its status free
end if
```

RELOCATABLE DYNAMIC PARTITIONS

Algorithm 6 Algorithm to delocate memory blocks - Job_location is between two free blocks

```
if job_location is between two free blocks then
    merge the three blocks
    mem_size(counter−1)=mem_size(counter−1)+job_size+mem_size(counter+1)
    mem_status(counter+1)=null entry
end if
```

RELOCATABLE DYNAMIC PARTITIONS

Algorithm 7 Algorithm to delocate memory blocks - Job_location is adjacent to another free block

```
if job_location is between two free blocks then
    . . .
else
    merge both blocks into one
    $\text{mem\_size(counter}-1) = \text{mem\_size(counter}-1) + \text{job\_size}$
end if
```

RELOCATABLE DYNAMIC PARTITIONS

Algorithm 8 Algorithm to delocate memory blocks - Job_location is isolated block

```
if job_locaton is adjacent to one or more free blocks then
    . . .
else
    search for null entry in free memory list
    enter job_size and beginning address in the entry slot
    set its status free
end if
```

RELOCATABLE DYNAMIC PARTITIONS

- Relocatable dynamic partitions
- Gather all free blocks
- Compact them into one block large enough to accommodate new job in-waiting
- Example

RELOCATABLE DYNAMIC PARTITIONS

- Analysis
  - Advantage
    - Eliminate wasted memory blocks
    - A new job can be loaded if it s size if not bigger than that of the free memory block
    - Difficulties in compaction
    - Relocate all programs so they are contiguous
    - Adjust every address and every reference to an address within each program
    - Data values must be left alone

SUMMARY

- Problem-solving driven
- Single-user scheme
- Fixed partitions
- Dynamic partitions
- Relocatable dynamic partitions