Operating System - P68

dynamic priority adjustment (动态优先级调整), to improve performance
- don't always know in advance which jobs are CPU bound or I/O bound
- behavior may change at different phases of execution
- usually want to give higher priority to I/O bound and interactive job
- dynamically adjust the priority based on current process behavior
  - process use entire time quantum without blocking
    - can be adjusted downward slightly in priority
    - repetition will gradually decrease priority
  - process block before using entire time quantum
    - can be adjusted upward based on fraction of time quantum used
    - repetition will gradually increase priority

SPN (shortest process next scheduling, 最短进程优先)
- predict the execution time of an interactive command
  - get optimal response time
- prediction value is used to run the estimated shortest process next
  - based on previous times using a decaying weighted average
    - a ($0<a<1$): weighting factor
    - T_i : measured execution time for execution i] $i \in N$
    - W_i : weighted average time after execution i]

$$\begin{cases} W_0 = T_0 \\ W_i = aW_{i-1} + (1-a)T_i, & i>0 \end{cases}$$

  - smaller a : recent behavior have more influence
  - larger a : past behavior have more influence
- combine shortest process next (SPN) with priority scheduling
  - predict to have shorter execution time with higher priority

guaranteed scheduling (保证调度)
- guarantee that each process will received particular fraction of available CPU time
  - system tracks CPU usage since process creation
  - compute ratio of time actually used to time guaranteed
  - run process with lowest ratio
  - recompute periodically (each scheduler run)

Operating System - P69

lottery scheduling (彩票调度), scheduler pick next process to run randomly
each process is assigned probability of being chosen
probability assignment similar to priority system

fair-share scheduling (公平分享调度), similar to guaranteed scheduling
except that CPU allocation is per user instead of per process

time plays essential role in scheduling in real-time system
category : hard real time (硬实时)
soft real time (软实时)
periodic and aperiodic
schedulable (可调度的) : m is the number of periodic events
C_i is CPU time required for event i
P_i is the length of the period of event i
schedulable $\iff \sum_{i=1}^{m} C_i/P_i \leq 1$

scheduling policy (调度策略) : importance level of each process
how to adjust priority
how to apportion CPU among user, process, thread, kernel
scheduling mechanism (调度机制) : implementation of the scheduling policy

memory manager (内存管理), keep track of occupied/available main memory
cache management done in hardware
register management done by compiler
allocate memory to process when need
tell process which part of free memory to use
deallocate memory when process done using
mark the memory area as available

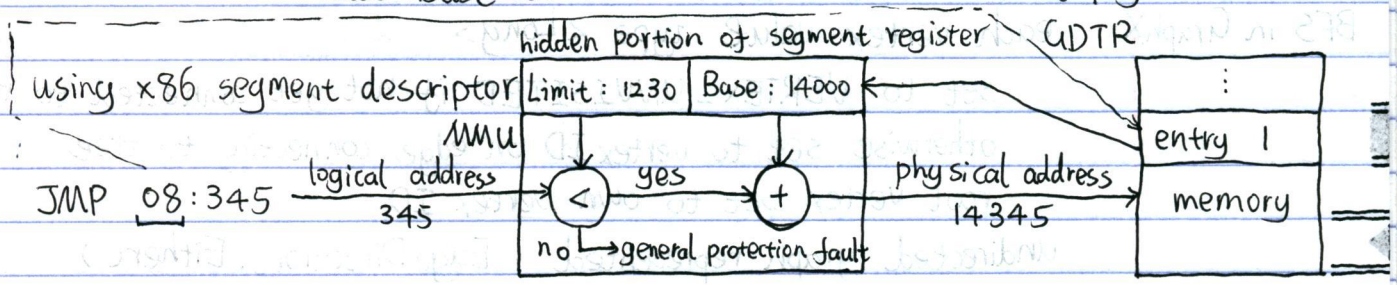
relocation problem (重定位问题)
program assume that the first instruction at memory address 0
if program placed at a location other than was assumed
all address based calculation fail

Operating System - P70

static relocation (静态重定位)
- before starting execution, modify all address in program by adding the actual start address of the program
- complicated since not all numbers are address

process address space (进程地址空间), set of addresses process can use
- static relocation complicated in software
- disconnect link between address in program instruction and physical memory
  - each process will have own address space
  - address x in one process different from address x in other
    - logical (virtual) address : address generated by program
  - each process created assuming : the only one running
    placed at physical memory address 0

dynamic relocation (动态重定位)
- memory management unit (MMU), hardware component doing address translation
  - simple MMU has two registers : Base, Limit
  - when program run : load Base with start address of program in physical memory
    load Limit with size of program
  - when refer to memory address m :
    MMU check if m less than the value in Limit
    add Base to m to determine the actual physical address



memory allocation (内存分配), main memory usually divided into two partition
- resident operating system, usually held in low memory with interrupt vector
- user process held in high memory
- keep track of free memory : quick allocation / de-allocation
- allocate memory area : contiguous / non-contiguous

Operating System - P71

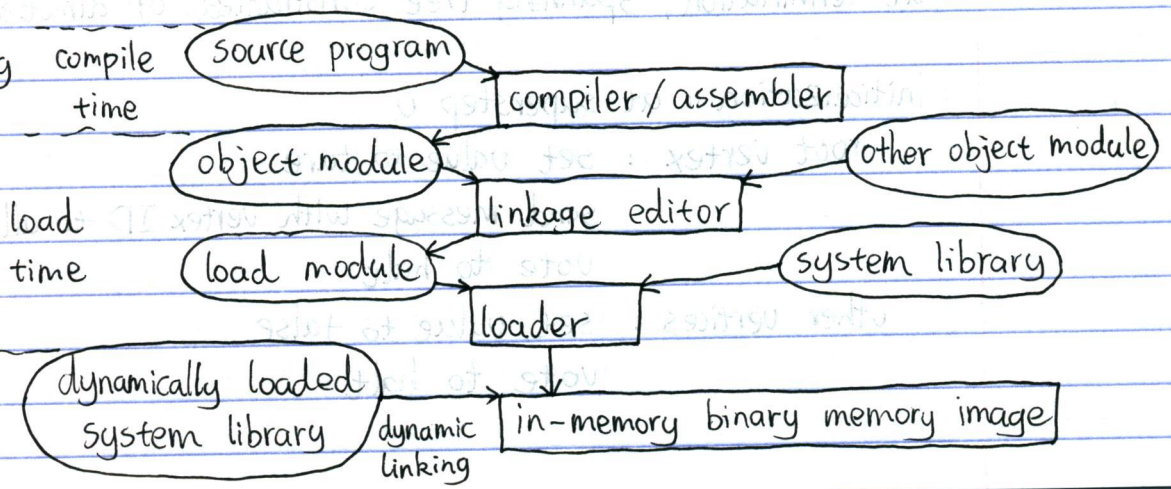
tracking memory usage with bitmap (位图)
divide memory into allocation units or frames
memory always allocated in multiple of allocation units

tracking memory usage with list (链表)

compaction (紧缩), all contiguous free areas less than frames process need
Compact memory by moving jobs
potentially time-consuming

address binding, giving each instruction/variable one memory address
program may be loaded in any part of physical memory
compiler : bind symbolic address to relocatable address
loader : bind relocatable address into absolute address
address binding done in different stage
compile time: absolute code can be generated if memory location known
must recompile code if starting location change
load time: address assigned while loading program into memory
must generate relocatable code at compile time
execution time: binding delayed until run time
if process can be moved during execution
need hardware support for address map

multistep processing



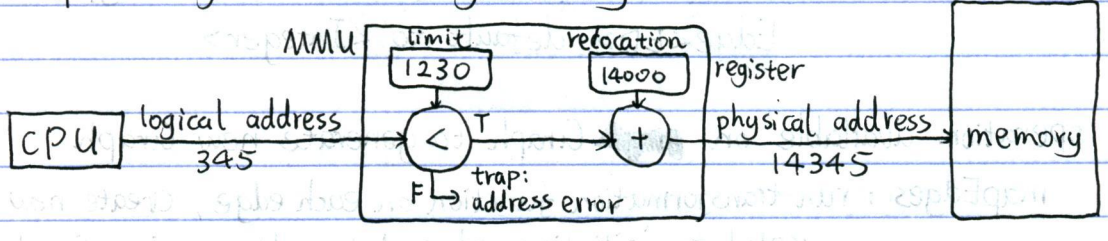
Operating System - P72

logical address (逻辑地址), generated by CPU, refered to virtual address
for the CPU, program is continguous array of memory address

physical address (物理地址), seen by the memory unit, real address on hardware
与 logical address : same in compile-time
and load-time address-binding scheme
different in execution-time scheme

linear address (线性地址), one level in x86 protected mode
generated by the CPU using the segment table entry
refered to virtual address instead of logical address
equivalent to physical address when not using paging

MMU (memory management unit, 内存管理单元), hardware :: logical address -> physical address
user program deal with logical address
never see real physical address
relocation register used to protect
user process from each other
and from changing operating-system code and data
register content : base (relocation) register, value of smallest physical address
limit register, range of logical address
logical address must less than limit register
map the logical address dynamically



partition memory divided into partition, at most one program in one partition
fixed partition : partition size fixed
variable partition : OS moniter parts of memory
which are occupied and which are available

Operating System - P73

allocation hole (free blocks of memory) of various size
scattered throughout memory
process allocated memory from contiguous block of memory
large enough to accommodate
OS maintain information about allocated patition and free partition (hole)

dynamic storage allocation: from hole
first fit (首次适配): first hole found
best fit (最佳适配): the smallest hole of size large enough
result in smallest leftover hole
worst fit (最差适配): the largest hole
result in largest leftover hole

external fragmentation (外部碎片)
total memory space exist to satisfy a request, but not contiguous
free memory space get broken into little fragmentation
as process loaded and removed

internal fragmentation (内部碎片)
allocated memory may be slightly larger than requested memory
when allocate fixed-size block
extra memory is internal to block and never used

compaction (紧缩) reduce external fragmentation
shuffle memory content to place all free memory together in one large block
possible only if relocation dynamic, done at execution time
use non-contiguous allocation
paging / segmentation
part of process can be different parts of memoy

32-bit Logical (virtual) address space

text	data	heap → ← stack	kernal address space
executable instruction	global variable	dynamically allocated memory	cannot touch

0x0 ... process address space ... 0xB...FF 0xF...FF

Operating System - P74

paging

frame : divide physical memory into fixed-size allocation unit
page : divide logical memory into unit of same size as frame
keep track of all free frames
for process of n frames, find n free frames and load process in the frame
page table : set up to translate logical to physical address

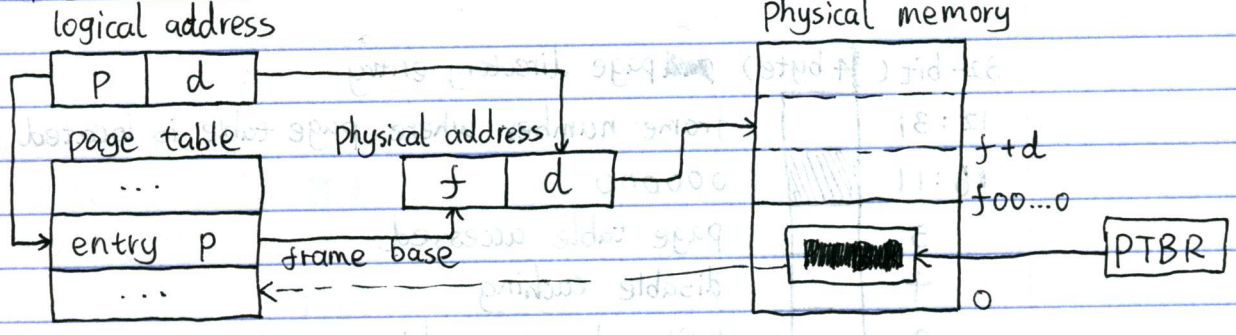
32-bit (4 byte) page table entry

12:31		frame number (20-bit)
9:11		000
8		global page
7		0
6		page modified (dirty bit)
5		page accessed (read bit)
4		disabe caching
3		write-through caching
2		accessible by all
1		write allowed
0		page present (present bit)

address translation scheme for given logical address space of 2^m and size of 2^n

logical address divided into | p (m-n bit) | d (n bit) |

page number (p) : use as index into page table
get base address of frame corresponding to page
page offset (d) : add to base address to determine physical address



PTBR (page table base register) : loaded with start physical address of page table
page table kept in main memory, PTBR point to page table

Operating System - P75

TLB (translation look-aside buffer, 转换检测缓冲区)

special fast-lookup hardware cache

associative, high speed memory

make faster: every data/instruction access require two memory addresses
one for page table, one for data/instruction

high speed parallel search

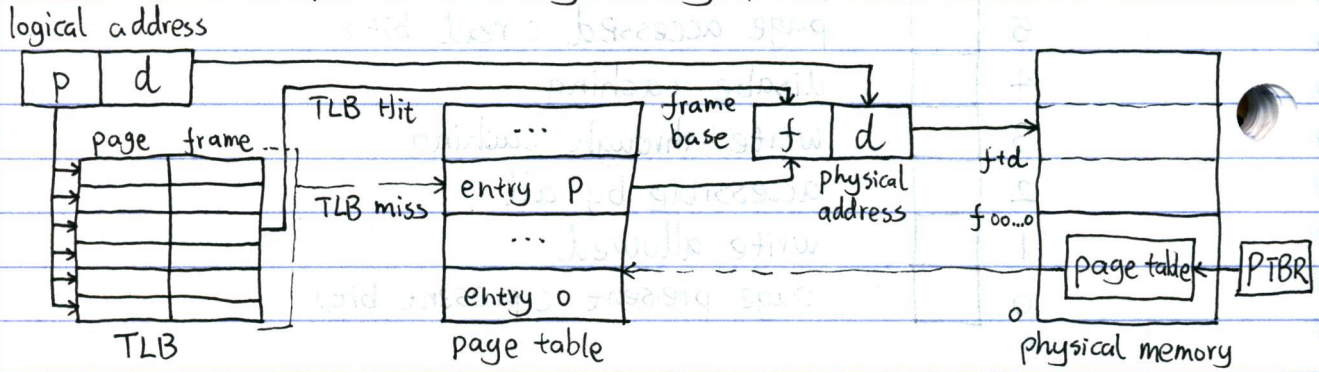
input: page number

lookup: parallel lookup of associative array for page number

output: frame number / TLB miss

frame reference must be obtained from page table
and the TLB must be updated

TLB flush/update necessary during process switch



two-level paging, logical address broken down to 3 part

P_1	the page directory entry
P_2	the index into the page table pointed by P_2
d	add to base address to determine the physical address

32-bit (4 byte) page directory entry

12:31		frame number where page table is located
6:11		000000
5		page table accessed
4		disable caching
3		write-through caching
2		accessible by all
1		write allowed
0		page table present

Operating System - P76

x86 paging model

32-bit paging	directory*	table 10-bit	offset 12-bit		4KB page
	directory 10-bit	offset 22-bit			4MB page

PAE paging	directory pointer	2-bit	directory 9-bit	table 9-bit	offset 12-bit	4KB page
		2-bit	directory 9-bit	offset 22-bit		4MB page

X86-64 paging	PML4	Dir. Ptr.	directory	table	offset 12-bit	4KB page
	PML4	Dir. Ptr.	directory	offset 21-bit		2MB page
	PML4	Dir. Ptr.	offset 30-bit			1GB page
	9-bit	9-bit	9-bit	9-bit	12-bit	

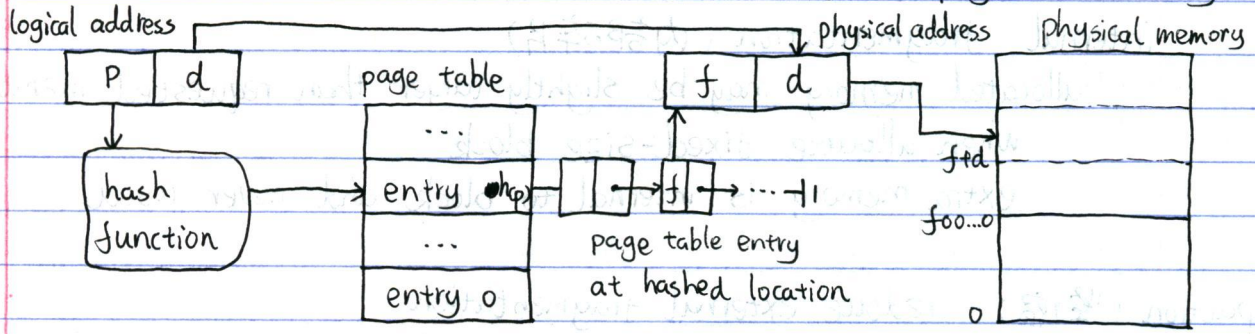
inverted page table (倒排頁表), one entry for each frame in memory

each entry consist of : page stored in the frame
process that own the page

decrease memory needed to store each page table
only one page table in system

increase time needed to search the table when page reference occurs

use hash table to limit the search to few page table entry

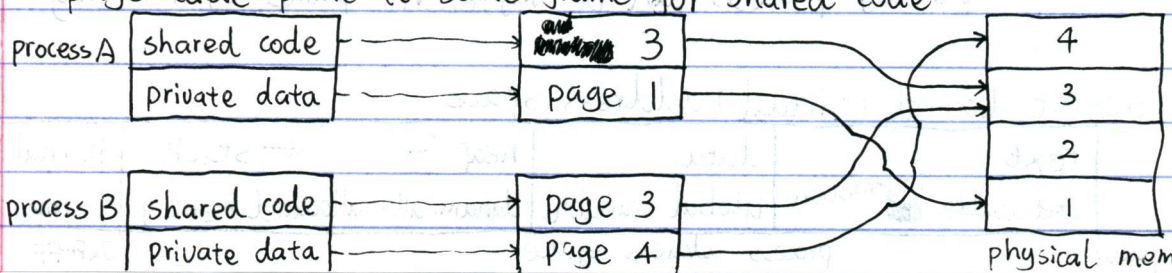


shared page

shared code : one copy of read-only (reentrant) code shared among processes

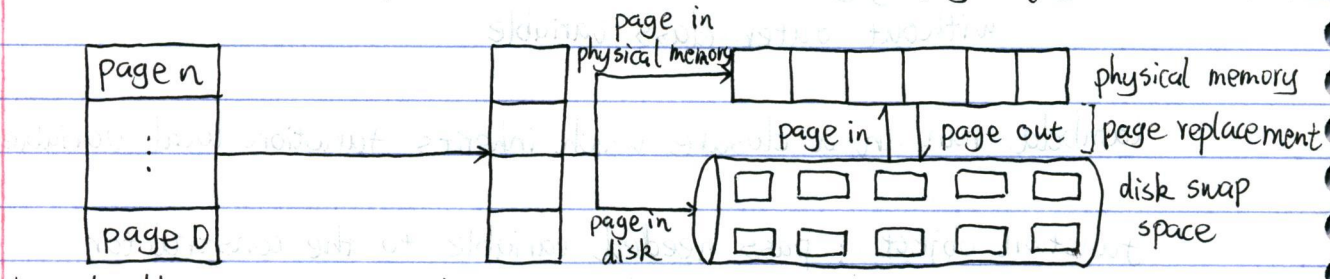
private data : each process keep separate copy of data

page table point to same frame for shared code

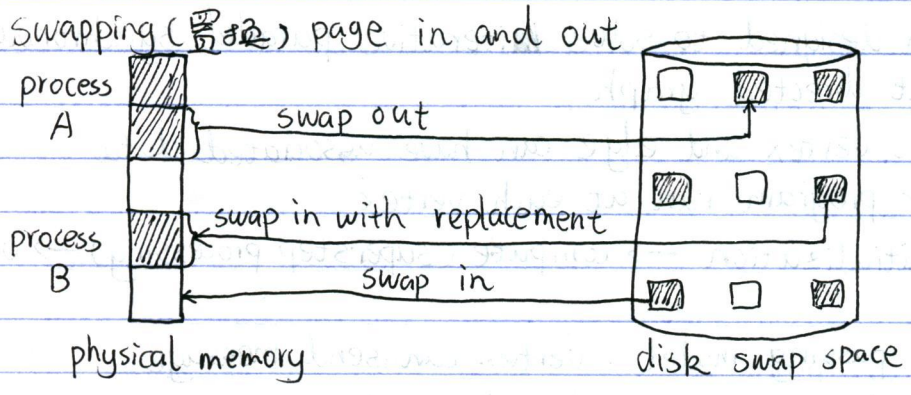


Operating System - P77

virtual memory : only part of program need to be in memory for execution
keep only some pages of the process in physical memory
other kept in special disk space (swap space)
using paging structure to indicate certain page not in physical memory
bring page in from swap space when needed
swap page out to swap space when low in memory
process used up maximum number of allowed physical frame



swapping (置換) page in and out

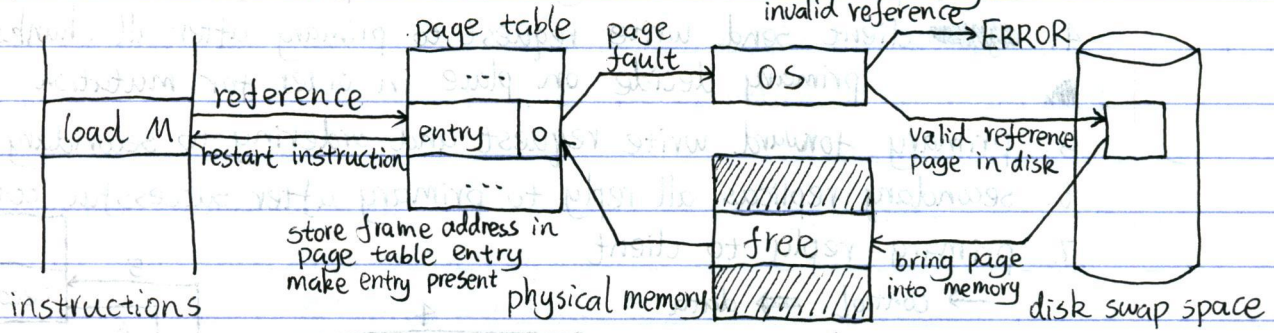


page fault : present bit : associated with each page table entry.
1 : in-memory , 0 : not-in-memory
during address translation, present bit in page table
1 : access frame 0 : page fault
mean trying to access page not in memory
can be because page not mapped / in swap space
exception thrown if page fault happen

when page fault address unmapped
operating system must decide allocate page or abort (segmentation fault)
Os can store swap space location in page table entry when page swapped out

Operating System - P78

page fault
find free frame (with/without replacement)
read in desired page into allocated frame
modify page table entry
restart the instruction cause the page fault
program counter point to faulting instruction
hardware must be able to do this automatically



no free frame, find some frame in memory not really in use, then swap out
need page replacement algorithm
should result in minimum number of page fault
same page may be brought into memory several times

basic page replacement
find location of desired page on disk
find free frame: if there is a free frame, then use
if no free frame, use page replacement algorithm
to select victim page
write victim page to disk if page modified
update victim page's page table entry
to indicate page in swap space (can store location here as well)
bring the desired page into the newly free frame
update page table
restart user process

reference string, string of page number accessed by process during execution
useful in computing number of page fault will occur
when particular replacement algorithm used