“Operating systems” course by Peter Lipp

Lukas Prokop

September 25, 2013

Contents

Some content taken from “Modern operating systems” (Tanenbaum)

Ring 0 (most privileged) to Ring 3 (least privileged). Ring -1 for VT-x. `int 0x80` is used to switch to kernel mode.

1 Process

Process is registered in process table. “Core image” is an instance of a process in memory. Process consists of address space, executed binary, program data, stack, UID of process owner, working directory, process children and the register set (program counter, stack pointer, hardware registers, …) …

There is no real difference between user- and kernel-threads, but kernel threads are running in kernel mode all the time and are part of the operating system (kernel). User threads are associated with user processes. Typically user threads do have their own scheduling algorithm and the kernel does not know anything about the user threads.

Copy-on-write: In the event of a `fork`, we mark program text and data as shared page for children and parent. Once a modification is about to take place, the page gets copied for the child. So copying is delayed until modification.

Multilevel Paging Architecture (more than 1 level) is useful, because you are not forced to keep all layer data in memory.

Zombie: A child process waiting for his parent to read its return value (which is always expected). A zombie will always be created whenever `waitpid` is not called.

2 Synchronization & Deadlocks

A set of processes is deadlocked if each process in the set is waiting for an event that only another process in the set can cause.

Deadlock handling using recovery through killing processes, recovery through rollback or recovery through preemption. Preemption is the act of temporarily interrupting a task carried out by a computer system, without requiring its cooperation and with the intention of resuming the task at a later time.

4 requirements for a deadlock:

- Circular wait (Circular dependency)
- Hold and wait (Process holding resources can request further resources)
- Mutual exclusion (only finite number of processes in critical region; number is smaller than number of resources; otherwise resource is free for allocation)
- No preemption (Process cannot be taken away a resource)

Fairness either refers to the requirement that the order of acquires is reserved for the execution ("first come, first serve") or CPU time is distributed uniformly (relaxed to the fact that each process must be assigned CPU time eventually in infinite time).

Race condition = "Behavior of a system where the output depends on the sequence order of execution"

2.1 Example for deadlocks

Computer with tape drive and CD recorder. Two processes need to produce a CD ROM from data read from the tape.

1. Process #1 requests and is granted the tape drive.
2. Process #2 requests and is granted the CD recorder.
3. Process #1 requests CD recorder and is suspended until process #1 returns it.
4. Process #2 requests the tape drive and is also suspended, because it's assigned to #1

2.2 Synchronization solutions

4 conditions for efficient solution:

- Not more than 1 process in critical region
- No assumptions about relative speed or number of parallel units
- No process outside of critical region is allowed to influence other processes
- No process has to wait infinitely to access critical region

2.3 Synchronization APIs

Semaphore:

- wait/signal
- wait/post
- V/P
- pend/post
- down/up

Mutex:

- lock/unlock
- acquire/release

Condition Variable:

- wait/signal

3 Page Replacement Theory

Trashing means a program is causing page faults every few instructions. So the CPU time is only spent on swapping pages and not on executing some task.

3.1 Bélády's anomaly

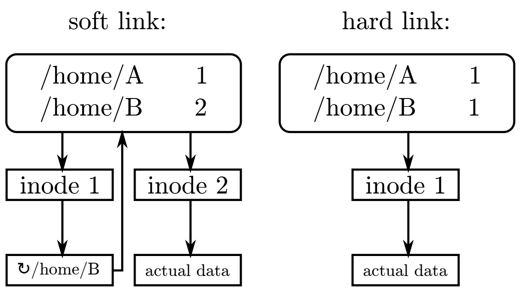
Intuition: More page frame slots in main memory means fewer page faults. Intuition is wrong. Shown for FIFO, 3 and 4 page frames and page sequence $\{0, 1, 2, 3, 0, 1, 4, 0, 1, 2, 3, 4\}$.

3.2 Optimal PRA

Proven by Bélády, 1960. Look into the future and label pages by the number of instructions executed before this page is used. Is proven to be optimal, but looking into the future is impossible (compare to LRU algorithm where we are looking into the past).

4 Filesystem

Inodes (according to POSIX) store the following information: file size, device ID, user ID, group ID, file mode, file content pointer, link count, ctime, mtime and atime. **NO** filenames. The relation filename to inodes is surjective. The relation file content to inodes is bijective.



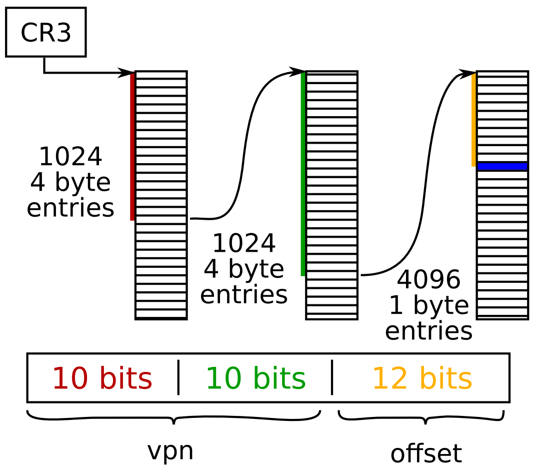
Internal fragmentation If you want to store 1 KB and use 4 KB pages, 3 KBs are unused and wasted.

External fragmentation Holes between two assigned data blocks are not large enough for new data block (reordering would result in a large enough slot).

5 Paging and addressing

5.1 x86 - Intel 32 bit architecture

32 bit addresses, typically 4KB pages. Multilevel Page Table with 2 levels. We use at least 1 page for program text, data and stack respectively. Why is the page size always a power of 2? Because the 12 offset bits of the virtual address can reference 2^n different bytes in one page. A physical page (physical address space) is called a page frame. In virtual address space it's called page. They are of the same size.



address	page size	number of pages
32 bit	4 KB page	1.05 mio. pages $= \frac{2^{32}}{4096}$
64 bit	4 KB page	68.7 billion pages
64 bit	2 MB page	134 mio. pages

Physical Address Extension 3 levels (additional “Page Directory Pointer Table”)

amd64 4 levels (PDPT is extended from 4 to 512 entries, additional “Page Map Level 4” (PML4))

At each level we store status information[1]:

Caching disabled bit For device registers relevant

Protection bit 0 = rw, 1 = rx (or actually use an rwx-bitmask)

Modified bit (“Dirty bit”) 1 = swap area page differs to main memory page, 0 = they equate

[1]Modified and Referenced bits are set by hardware; everything else is subject to operating system

Valid	VPN	Mod.	Protection	PPN
1	0110	1	RW	110
1	0111	0	RX	111
		...		

Referenced bit 0 = page has not been accessed, 1 = has been (relevant for PRA)

Present bit 1 = Page is in main memory available, 0 = N/A

If a page fault occurs...

1. Swap little-used page frame
2. Write contents back to disk
3. Fetch referenced page into free page frame
4. Update map

Translation Look-aside buffer (TLB, is an associative memory):

Input: Virtual address

Output: (maybe) Physical address

5.2 amd64 - 64 bit architecture

Currently only 48 of the 64 bits are in use (256 TB RAM addressable). Future specifications might extend the address to 52, 56 or the full 64 bits (16 EB TAM addressable).

6 Page Replacement Algorithms

6.1 Not Recently Used

Referenced & Modified bits are set by hardware. Periodically reset the reference bit. On pagefault classify into 4 categories:

Class 0 Not referenced, Not modified

Class 1 Not referenced, Modified

Class 2 Referenced, Not modified

Class 3 Referenced, modified

Remove page of lowest-numbered non-empty class.

6.2 FIFO

First In, First Out. Remove page that persisted the longest time in memory. Use list for storing page order. If no page fault (just access), do *not* put at list head.

6.3 Second Chance

Like FIFO, but if the oldest page has the R-Bit set, the bit is cleared and added to the buffer like new pages. The search goes on.

6.4 Clock

Pages are arranged in circular list. A hand points to the oldest page. On page fault, the page pointed to by the hand is considered. If R-Bit is not set, evict the page. Else clear R-Bit and advance hand. Only implementation-level difference to Second Chance.

6.5 LRU

Least Recently Used removes LRU page perferably. We need an efficient implementation for the counters. n pages ⇒ $n \times n$ matrix. page k is referenced, then set all bits of row k to 1 and all bits of column k to 0. The row with the smallest number is swapped out.

6.6 Aging

Software implementation of LRU. n pages ⇒ $n \times n$ matrix or bits. $\{r_0, r_1, \dots r_{n-1}\}$ are the referenced bits of the pages. Shift values in the matrix to the right and insert R-Bit on the left. Remove page with the lowest label number.

6.7 Working Set

A Working Set is the set of pages currently used by a process. WS clock tries to preload the working set in memory when the process is in charge. Prepaging means (contrary to Paging On Demand) loading pages before letting processes run. $w(k,t)$ = working set at time t with k most recent memory references. w is a monotonically, nondecreasing function with finite upper bound. Each process is assigned a working set of k pages. Prepage those pages. Now more specifically, working set = set of pages referenced in the past t seconds of virtual time. Periodic clock interrupt resets R-Bit. Hardware sets R- and M-Bit. On page fault:

- if page.R is 1, page.time = current virtual time
- if.page.R is 0 and (virtual time - page.time) $> \tau$, remove page from memory
- if.page R is 0 and (virtual time - page.time) $\leq \tau$, remove page with the smallest page.time (thus, oldest)

6.8 WS Clock

Based on Clock (circular list/ring) and working set (set of pages of process). On page fault, examine page hand is pointing to:

- if page.R is 1, page.R = 0; advance to next page
- if page.R is 0 and (virtual time - page.time) $> \tau$:
  - if page.M is 0, use this slot for the new page
  - if page.M is 1, schedule write to disk but still advance to next page

7 Virtualization

Full Virtualization, Partial virtualization, Paravirtualization.

We need some concept to map the virtual addresses of the guest system to the physical addresses of the host system. The MMU (Memory Management Unit) does not provide any support.

Shadow Table If guest changes his address space assignment, then notify Virtual Machine Monitor (VMM) and update shadow table.

- Pagefaults are handled by VMM
- Guest page table has to be kept up to date
- Emulate M- and R-Bits for guest system
- Up to 3 page faults for one guest page fault

Extended Page Table (Intel) or

Nested paging (AMD) One big TLB for all VMs. EPT tables are specifically for translating physical guest addresses (= virtual host addresses) to physical host addresses. Thus EPT base pointer is put into CR3 register in host system.

Page Walk Cache Store PTEs of all levels (except L1)

Kernel versus User mode: Virtualized system runs in user mode, but requires kernel mode. Thus, VMM runs in special ring “-1”.

“Virtualization allows us to represent resources of a computer in such a way that you can use them without knowing their properties.”

8 Software-based lock implementations

threadId $\in \{0, 1\}$, other $= 1 -$ threadId

Listing 1: Dekker’s algorithm

```
def init()
    flag[0] = 0
    flag[1] = 0
    turn = 0
```

```
def acquire()
    flag[threadId] = 1
    while (flag[other] == 1)
        if (turn == other)
            flag[threadId] = 0
            while (turn == other) busy_wait()
            flag[threadId] = 1

def release()
    turn = other
    flag[threadId] = 0
```

Listing 2: Peterson’s algorithm

```
def init()
    flag[0] = 0
    flag[1] = 0
    turn = 0

def acquire()
    flag[threadId] = 1
    turn = other
    while (flag[other] == 1 and turn == other)
        busy_wait()

def release()
    flag[threadId] = 0
```