Real time CPU scheduling

Priority-based scheduling

1)rate monotonic scheduling: shorter period with higher priority

2)earliest deadline first sch(EDF) (每个 ddl 检查一遍，交替执行)

3)proportional share sch: T shares

Page replacement

(1)FIFO (2)LRU a.age b.counter increment c.stack

(3)OPT (4)LRU approximation a.reference bit b.second-chance (5)counting algthm: a.LFU(最小) b.MFU(6) page-buffering: pools of free frames, modified pages list, intact

algorithm evaluation

1)deterministic modeling: analytic eval

2)queueing models (Little's formula)

3)simulation

4)implementation

S is serial portion

N processing cores

$$speedup \le \frac{1}{S + \frac{(1-S)}{N}}$$

1.**PCS**: process-contention scope: thread 在 process 里抢资源

2.**SCS**: system-contention scope

3.**Interrupt latency** - time from arrival of interrupt to start of routine that services interrupt

4.**Dispatch latency** -time for scheduler to take current process off CPU and switch to another

5.**STBR**: segment-table base register: segment 的表格

6.**STLR**: segment-length register

7.**PTBR**: page-table base register

8.**PTLR**: page-table length register

9.**TLB**: translation look-aside buffers/associative memory

10.**ASIDs**: address-space identifies: TLB 中标明每个 process 的 entry，不用在 context switch 的时候 flush

11.**Allocation of the frame**: (1)fixed allocation (2)proportional allocation (3)priority allocation (4)local/ global allocation (5)NUMA

12.**Kernel memory allocation**: (1)buddy-system: contiguous allocation (2)slabs: kernel object->caches->slabs

13.**Virtual memory** can be implemented via: (1) demand page (2)demand segmentation

14.**Unified buffer cache**: A unified buffer cache uses the same page cache to cache both memory mapped pages and ordinary file system I/O to avoid double caching (mem-mapped I/O + I/O using r/w)<->buffer cache<->fs (原来有 page cache)

15.**Log structured**(or **journaling**) file systems record each metadata update to the file system as a transaction

16.**CPU interrupt-request line**: triggered by I/O device and checked by processor after each instruction

17.**Computer system**: (1)hardware (2)OS (3)application programs (4)users

18.**AMP**: asymmetric multiprocessing: boss processer 控制系统， 其它处理器向 boss 要任务或者做预先的任务

SMP: symmetric multiprocessing:每个处理器参与完成 OS 的所有任务，processor 没主从关系。每个处理器都有自己的寄存器集，也有私有本地缓存，但共享物理内存

19.**NUMA(UMA)**: non-uniform memory access; cpu 访问 RAM 所需的时间不同

20.**dual mode**: user and kernel mode (1) mode bit

21.multiprocessor must provide **cache coherency**

22.**syscall**: a.advantages (1)portability (2)ease of use b.system-call interface(a table indexed accords to number

23.**scheduling queues of processes**: (1)job queue (2)ready queue (3)device queue

24. **rendezvous**: both send and receive are blocking in IPC

25. multicore == multiprocessor vs multicpu

26. thread-safe routines: system calls or library which can be called from multiple thread simultaneously&correctly

27.**homogeneous processor**: 每个 core 相同指令集

28.**SMP**: keep all CPU loaded for efficiency: (1)load balancing (2)push(pull): periodic check loads of each CPU

29.**soft/hard real-time system**: conflict phase of dispatch latency: (1)preemption of any process running in kernel mode (2) release by low-priority process needed by high-priority processes

30.**address binding** of instructions and data to memory addr: (1)compile time: abs (2)load time: relocate (3)execution time: binding delay until run time if a process be moved from one memory segment to another

31.logical and physical memory addr are same in compile and load time scheme; differ in execution-time scheme

32.**roll in/roll out**: swap lower-priority process for priority-based sch algorithm

33.**MMU**: memory-management unit

34**layer file system**: application prog->logical file sys->file-org module->basic file sys->I/O control->device

35.**File system implementation**: (1)Boot control block (2)Volume control block(superblock) (3)directory structure (4)File control block (4)Root partition (5)Virtual File system(VFS): interface of diff file sys; function table

36.**directory implementation**: (1)linear list (2) Hash table

37.**allocation method**: (1)contiguous: a.compaction off/on-line b.extend-base (2)linked allocation: file-allocation table(FAT) :存 link list 的 block (3)indexed

38.**free-space magnt**: (1)bit map (2)linked list (3)grouping(类似 index 存 free frame) (4)counting: contiguous allocation (5)space map: divide device space into metaslab associated with space map; in log with counting

- CPU scheduling decisions may take place when a process:
  1. Switches from running to waiting state
  2. Switches from running to ready state
  3. Switches from waiting to ready
  4. Terminates
- Scheduling under 1 and 4 is **nonpreemptive**
- All other scheduling is **preemptive**

Can be done by using the length of previous CPU bursts, using exponential averaging

1. t_n = actual length of n^{th} CPU burst
2. τ_{n+1} = predicted value for the next CPU burst
3. $\alpha, 0 \le \alpha \le 1$
4. Define : $\tau_{n+1} = \alpha\, t_n + (1-\alpha)\tau_n$.

Commonly, α set to ½

Effective Access Time (EAT)

EAT = (1 – p) x memory access
+ p (page fault overhead
+ swap page out
+ swap page in
+ restart overhead
)

Page Fault Rate $0 \le p \le 1$

Associative Lookup = ε time unit

Can be < 10% of memory access time

Hit ratio = α

Hit ratio – percentage of times that a page number is found in the associative registers; ratio related to number of associative registers

Effective Access Time (EAT)

$$EAT = (1+\varepsilon)\,\alpha + (2+\varepsilon)(1-\alpha)$$
$$= 2 + \varepsilon - \alpha$$

Information associated with each process

(also called **task control block**)

- Process state – running, waiting, etc
- Program counter – location of instruction to next execute
- CPU registers – contents of all process-centric registers
- CPU scheduling information- priorities, scheduling queue pointers
- Memory-management information – memory allocated to the process
- Accounting information – CPU used, clock time elapsed since start, time limits
- I/O status information – I/O devices allocated to process, list of open files

- n = average queue length
- W = average waiting time in queue
- λ = average arrival rate into queue
- Little' s law – in steady state, processes leaving queue must equal processes arriving, thus
  $n = \lambda \times W$

39.**interrupt com**: (2)Interrupt handler receives interrupts(maskable to ignore or delay some interrupts) (3)Interrupt vector to dispatch interrupt to correct handler a.Context switch at start and end b.Based on priority c.Some nonmaskable e.Interrupt chaining if more than one device at same interrupt number

40.**kernel I/O subsystem**: (1)scheduling (2)buffering: store data in mem while transferring between devices (3)caching (4)spooling: hold output for a device (5)device reservation: provide exclusive access to a device

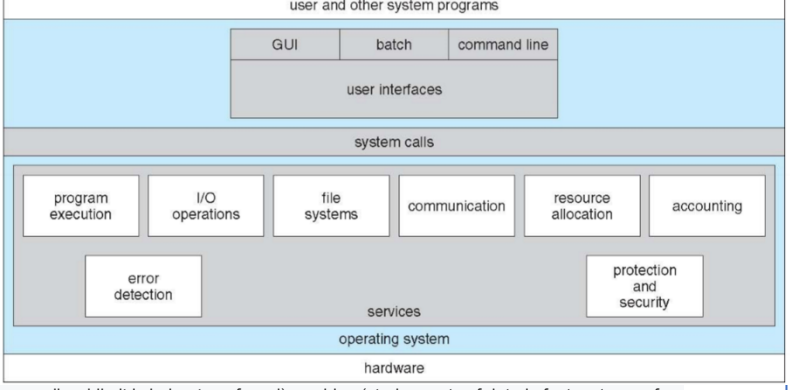
1.**Interrupt**

1)Interrupt transfers control to the interrupt service routine generally, through the interrupt vector, which contains the addresses of all the service routines. (hardware)

2)Interrupt architecture must save the address of the interrupted instruction

3)A trap or exception is a software-generated interrupt caused either by an error or a user request

4)An operating system is interrupt driven



2. The operating system preserves the state of the CPU by storing registers and the program counter

3. **IO Subsystem**: One purpose of OS is to hide peculiarities of hardware devices from the user.

I/O subsystem responsible for: 1)Memory management of I/O including buffering (storing data temporarily while it is being transferred), caching (storing parts of data in faster storage for performance), spooling (the overlapping of output of one job with input of other jobs) 2)General device-driver interface 3)Drivers for specific hardware devices

4.**Emulation** used when source CPU type different from target type (i.e.PowerPC to Intel x86)

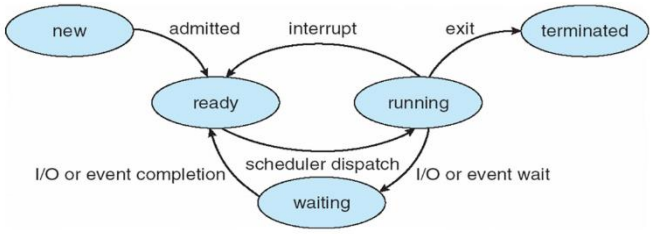
Generally slowest method

When computer language not compiled to native code **Interpretation**

5.**Virtualization** - OS natively compiled for CPU, running **guest** OSes also natively compiled

Consider VMware running WinXP guests, each running applications, all on native WinXP host OS

VMM provides virtualization services



6.**System Boot**: When power initialized on system, execution starts at a fixed memory location. Firmware ROM used to hold initial boot code. Bootstrap program typically stored in ROM or EFROM, which loads os kernel and start execution.

7.**Operating system must be made available to hardware so hardware can start it**

Small piece of code - bootstrap loader, stored in ROM or EEPROM locates the kernel, loads it into memory, and starts it

Sometimes two-step process where boot block at fixed location oaded by ROM code, which loads bootstrap loader from disk

8.Common bootstrap loader, GRUB, allows selection of kernel from multiple disks, versions, kernel options

9.Kernel loads and system is then running

20.**Types of system calls**

1)Process control

2)File management

3)Device management

4)Information maintenance

5)Communications

6)Protection

21.**Process scheduling**

(1)FCFS

(2)SJF--SRTF(preemptive)

(3)Priority scheduling (starvation-aging)

(4)Round Robin(RR): q>context switch (time quantum)

(5)Multilevel queue--Multilevel feedback queue

RR higher average turnaround than SJF but better responds

10.**system programs**

(1)File manipulation (2)Status information sometimes stored in a File modification (3)Programming language support (4)Program loading and execution (5)Communications (6)Background services (7)Application programs

12.**Programs**:(1)batch: jobs (2)time-shared: user programs/tasks

13.**system calls**: programming interface to the service provided by the os

14.Three general methods used to **pass parameters to the OS**

1)Simplest: pass the parameters in registers

In some cases, may be more parameters than registers

2)Parameters stored in a block, or table, in memory, and address of block passed as a parameter in a register

This approach taken by Linux and Solaris

3)Parameters placed, or pushed, onto the stack by the program and popped off the stack by the operating system

Block and stack methods do not limit the number or length of parameters being passed

11.**debuggung**

1)OSes generate log files containing error informatior (2)Failure of an application can generate core dump file capturing memory of the process (3)Operating system failure can generate crash dump file containing kernel memory

Beyond crashes, performance tuning can optimize system performance Sometimes using trace listings of activities, recorded for analysis Profiling is periodic sampling of instruction pointer to look for statistical trends

15.**IPC**: (1) shared memory (2)message passing: a.Direct communication b.Mailbox

16.**Mutithreading models**: (1)1:1 (2)1:N (3)N:M methods: (1)thread pools (2)openMP (3)grand central dispatch

18.**scheduler activations**: M:M and two-level needs light wight process(LWP) between kernel and user.

upcall communicate between kernel to the upcall handler in the thread library

19.**Convoy effect**: short process behind long process

20.**interrupt handling**: (1)the os preserves the state of the CPU by storing registers and the program counter

(2)determine which type of the interrupt: polling or vectored interrupt system

(3)separate segments of codes determine the actions for interrupt

22.**device driver**: for each device controller to manage I/O and provides uniform interface between controller and kernel

23. **DMA**: device driver transfer blocks of data from buffer storage directly to main memory without CPU intervention

24.**dynamic storage allocation**: (1)best-fit (2)first-fit (3)worst-fit

25.**multistep processing of user program**: source prog-(compiler/assembler)->object module-(linkage editor + other object modules)->load module->loader(+system library: linker)->in-memory binary memory image(+dyn loaded mem img)

26.**static linking**: system libraries and program code combined by the loader into the binary program image

dynamic linking: stub used to locate the appropriate memory-resident library routine

27.Thrashing: (1)working set(WSS) (2)page-fault frequency

28.other issues: (1)prepaging (2)page size (3)TLB search (4)I/O interlock

29.**file** lock: shared/exclusive lock 30.**access method**: (1)sequence (2)direct access (3)index

directory: (1)single-level (2)2-level (3)tree (4)acyclic-graph (5)general-graph: no cycle: only links to file/garbage collection/cycle detection algtm

21. **I/O structure**: Synchronous I/O: After I/O starts, control returns to user program only upon I/O completion (1)Wait instruction idles the CPU until the next interrupt (2)Wait loop (contention for memory access) (3)At most one I/O request is outstanding at a time, no simultaneous I/O processing

Asynchronous I/O: After I/O starts, control returns to user program without waiting for I/O completion (1)System call - request to the OS to allow user to wait for I/O completion (2)Device-status table contains entry for each I/O device indicating its type, address, and state (3)OS indexes into I/O device table to determine device status and to modify table entry

31.**page table structure**: (1)hierarchical paging (2)Hashed page table (3)inverted page table