TNE30019/TNE80014 – Unix for Telecommunications

Operating Systems Basics

Dr. Jason But

Swinburne University

Outline

- Multi-Tasking
- Memory Management
- Kernel vs. User space

Multi-Tasking

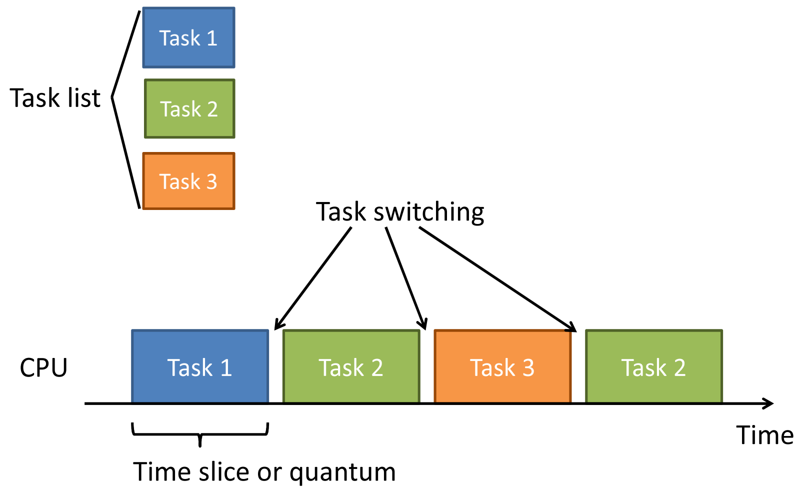
Older computers typically had 1 CPU (1 core)

- Can only run one job at a time
- One program counter
- One memory block
- One CPU Cache

How do we multi-task?

- **Preemptive** task switching is common
- Run one task for a while and then...
- Interrupt and suspend current task
- Run another task

Multi-Tasking – Task Switching



Multi-Tasking – Task Switching

Periodically

- Stop CPU
- Save current program's context
- Load next program's context
- Start CPU

Program Context

- Program counter
- CPU registers
- Program stack memory

Multi-Tasking – Switching Frequency

Configure task switching frequency

- 100Hz to 1000Hz common default
- Configurable with kernel parameter
- Set parameter in /boot/loader.conf (FreeBSD)
- Recompile kernel (Linux)

- Advantages/disadvantages of low frequency?
- Advantages/disadvantages of high frequency?

Multi-Tasking – Interrupts/Exceptions

Some tasks demand immediate attention

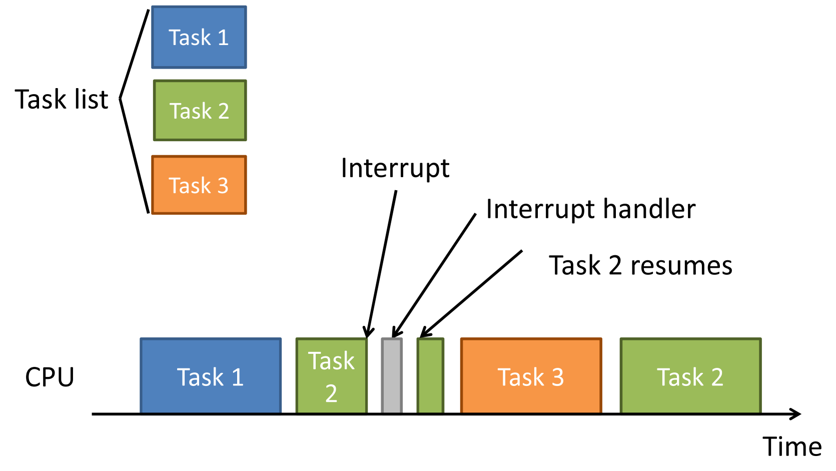
- Hardware has received new data
- Errors, e.g. division by zero
- Timer events
- Device or CPU triggers hardware or software **interrupt**

CPU is interrupted

- Pause current task
- Execute small interrupt handler
- Task is restarted after interrupt function completed

- Task-switch realised as periodic interrupt that changes context to new task

Multi-Tasking – Interrupts



Multi-Tasking – Task States

Running Task

Task currently executing, only one task can run on each CPU core

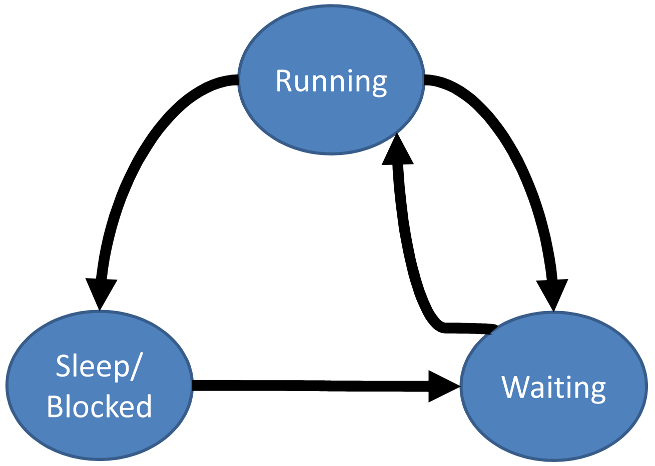
Waiting Task

Task can run, but is not running

Sleeping/blocked Task

- Task wants to wait for some time
- Task cannot run because it is waiting for another task to finish first

Multi-Tasking – Task States



Multi-Tasking – Which Task?

Where is task information stored?

Task list also known as run queue

Who chooses which task to run?

Operating system **scheduler**

How does the scheduler choose?

- Task state – running, waiting, sleeping/blocked
- Specified task priority – highest priority first
- Task run-time – how much CPU time task already had

Memory Management

Virtual Memory

- Pretend there is more memory than there really is
- Actual RAM is window into larger virtual memory space
- Virtual RAM stored on disk

To access memory not in current RAM Window

- Memory page from RAM is updated back to disk
- Required page from disk is loaded into RAM

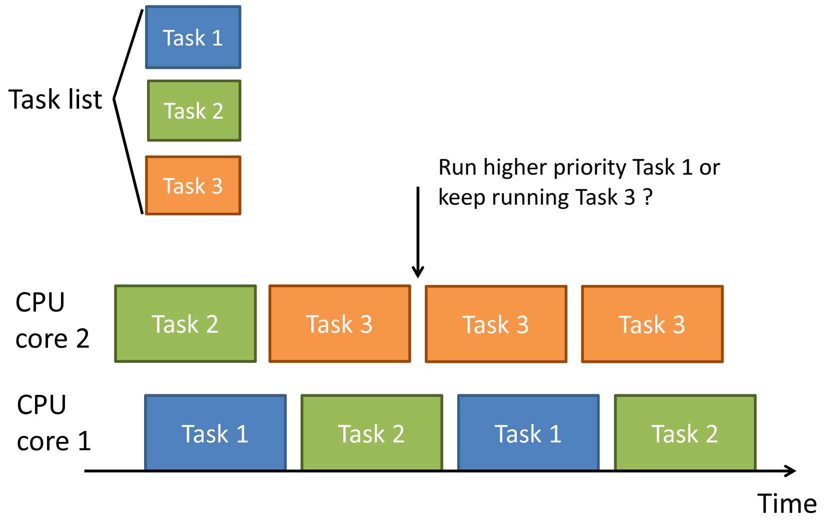
Layers of caching

1. CPU micro-cache cached recently calculated values
2. CPU registers cache frequently needed variables
3. CPU L1 cache caches most common portion of L2 cache
4. CPU L2 cache larger – caches common portion of RAM
5. RAM – caches virtual memory on disk

Multi-Tasking – Multi-core Systems

- Task-switching complicated on systems with more than 1 core (or more than 1 CPU)
- True parallel execution
- Shared memory, shared code-space, and shared devices
  - Need synchronization to avoid race conditions
- Per CPU core registers, L1 and L2 cache
  - Migrating tasks between CPU cores reduces cache efficiency
  - Scheduler needs to optimize use of available CPU cores

Multi-Tasking – Multi-core Systems



Kernel Space vs. User Space

Kernel

- Core of operating system, runs in kernel space
- Full access to memory and all other hardware
- Trusted, well-tested code (hopefully)

User processes (including root's processes)

- Run in user space
- Limited memory access
- Access to system resources via **system calls**
- System call will trigger special interrupt
- This interrupt will execute kernel system call code and return results to user process