FACULDADE DE CIÊNCIAS E TECNOLOGIA
DEPARTAMENTO DE INFORMÁTICA

Fundamentos de Sistemas de Operação

LEI - 2023/2024

Vitor Duarte
Mª. Cecília Gomes

1

Aula 22

- Tipos de SO
- Estrutura do SO
  - Monolíticos vs *Microkernels*
- Máquinas virtuais e hipervisores

- OSTEP: cap. B
- Silberschatz, Operating Systems Concepts, 10th Ed, Cap 2.7, 2.8
- artigo: The Evolution of an x86 Virtual Machine Monitor, Ole Agesen, et al. ACM SIGOPS Operating Systems Review, vol 44, 4, December 2010 - https://doi.org/10.1145/1899928.1899930

2

Vários SO para várias utilizações

Infraestruturas:

- Mainframes
- Servidores
- PC/portáteis
- SoC e micro controladores
  - Telemóveis
  - TV
  - Controlo industrial
  - Etc...

Requisitos:

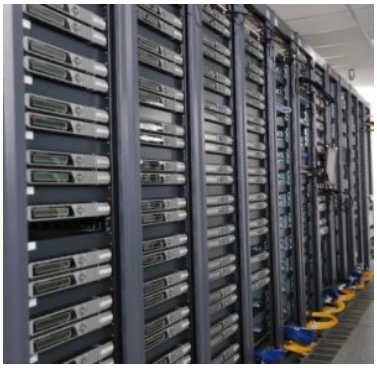
- Multiprocessadores
- Embutidos/firmware
- Serviços Online
- GUI
- Sistemas de tempo real

Mainframes

- Grandes arquiteturas, executando um SO
  - Grandes recursos (Ram, disco, bandwith de rede e IO)
  - Hardware especializado, tolerante a falhas e *hot-swappable*
  - Executam muitas tarefas *batch* e outras de atendimento concorrente de pedidos do exterior (OLTP)
- O SO tem de lidar com esses grandes recursos, com o *hot-swap* e optimizar o IO para discos e com o exterior
- Exemplos: IBM OS/390, IBM z/OS, ou do tipo Unix

Servidores

- Grandes e médios sistemas, ou grupos (clusters) cada um executando o seu SO
  - Grandes recursos (Ram, disco, bandwith de rede e IO)
  - Pode ou não ter hardware tolerante a falhas e hot-swappable
  - Executam muitos processos que oferecem serviços ao exterior (atendem pedidos concorrentes) e interações com o grupo (via rede)
  - Serviços de coordenação/partilha do cluster (loadbalancing, replicação, eleição master, etc.)
- O SO tem politicas para lidar com esses recursos, e optimizar o IO para discos e com o exterior.
  - Executa muitos processos dos serviços ou para interação com os restantes sistemas
- Exemplos: do tipo Unix, MS-Windows Server

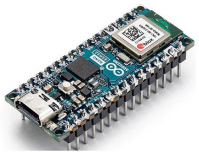


PC/portáteis

- Pequenas arquiteturas ou médias (workstations)
  - Pensado em servir um ou poucos utilizadores em simultâneo
  - Poucos/médios recursos (RAM, disco). IO limitado
  - Executam muitos processos, mas com diferentes importâncias
  - Normalmente oferecem interfaces gráficas
- O SO tem politicas para optimizar a interação com um utilizador.
  - Obter bons tempos de resposta na interface gráfica. Lidar com aplicações com requisitos temporais como as multimédia
- Exemplos: do tipo Unix (ex OS/X), MS-Windows

SoC e micro-controladores

- Pequenas arquiteturas (exemplo System-on-a-Chip)
  - Pensadas para pequenos dispositivos: TV, telemóvel, relógios, sistemas IoT, etc. Ou para estarem embutidas noutro equipamento (p.e. controlo industrial)
  - Hardware fixo de fabrica e com poucos recursos (RAM, disco) e podem ter bateria limitada. Podem não ter modos de execução ou MMU
  - Executam poucos processos (ou só um), podendo nem ter interface direta com utilizador
- O SO tem politicas para lidar e poupar esses recursos
  - Pode ter objetivos de poupança de energia, de tempos de resposta (real-time), fiabilidade
- Exemplos: alguns do tipo Unix (ex. IOS, Linux/Android), QNX, VxWorks



NOVA FACULDADE DE CIÊNCIAS E TECNOLOGIA DEPARTAMENTO DE INFORMÁTICA

7

Arquiteturas multiprocessador

- Múltiplos processadores (cores)
  - Com ou sem memória partilhada (clusters)
  - Permitem aumentar o processamento
  - Aparecem, em vários graus, dos mainframes aos mini sistemas
- O SO tem de gerir esses CPU para obter o desempenho correspondente
  - Lidar com problemas de concorrência interna no SO
  - Escalonador tem de manter todos os CPU ocupados, da melhor maneira

NOVA FACULDADE DE CIÊNCIAS E TECNOLOGIA DEPARTAMENTO DE INFORMÁTICA

8

SO em multiprocessadores

- Computadores com múltiplos CPU mas uma só memória:
  - Várias implementações são possíveis, a ideal é SMP – *Symmetric MultiProcessing*
  - O SO é um mas pode executar em qualquer CPU
    - Permite vários threads internos: controlo de concorrência interna mais complicado
  - O escalonador deve ter politicas de aproveitamento de todos os CPU
    - Executar qualquer processo em qualquer dos CPU?
    - Com multiplos threads por processo, como distribuir os seus threads pelos CPU?

Sistemas embutidos (*embedded*)

- Sistemas que fazem parte do hardware (podem ser chamados de firmware)
  - São normalmente SoC ou micro-controladores, podendo não ter MMU ou modos de execução. Mais recursos significam dispositivos mais caros
  - Distribuídos com o equipamento para o seu funcionamento e não acessíveis ao utilizador para além do que está definido pelo fabricante
  - O software está à partida definido e controlado pelo fabricante
  - Pode ter requisitos de tempo real

- O SO minimiza o uso de recursos
  - Adaptado à arquitetura e configurados para incluir apenas o necessário e minimizar uso de recursos
  - Todo o software (SO e aplicação) pode ser um "programa" fixo de fábrica
- O sistema+aplicação é desenhado para um único fim: o funcionamento do dispositivo
  - Não há necessidade de lidar com expansão do hardware ou novos programas, utilizadores, etc.
  - O SO não é para uso geral (não há vários programas ao gosto do utilizador). Tudo pode ser um só "programa"
  - Optimizados em conjunto para cumprir o fim a que se destinam
  - O escalonamento pode estar fixo, suportar tempo-real, ou não existir

Sistemas de tempo-real

- Requisitos temporais têm de ser cumpridos
  - Exemplos: controlo na industria, aviação ou automóvel, sistemas multimédia (alguns são micro-dispositivos)
  - Tempo de resposta a eventos (externos ou internos) é mais ou menos crítico
    - Mais crítico → ***hard realtime***
    - Menos crítico → ***soft realtime***
- O SO tem de poder garantir a resposta em certos instantes para certos processos

Escalonamento com tempo-real

- Os processos RT são escalonáveis se garantirmos que a soma dos tempos de processamento, para todos os eventos que podem ocorrer, cabe no tempo disponível de CPU
  - Estes processos apresentam normalmente um perfil tipo IO bould
- O SO e as aplicações podem ser um único "programa" num loop a atender eventos, para diminuir os overheads
- Soft real-time: as latências admitidas são maiores e aceitam-se atrasos na resposta a eventos
  - Pode ser conseguido apenas por escalonamento com maior prioridade para alguns processos face a outros
  - Alguns podem não ter limite de time-slice (nonpreemptable) e/ou terem prioridade fixa
- Hard real-time: os constrangimentos temporais têm de ser garantidos
  - Para além do anterior, tendo informação do tempo de cada execução, o escalonador ordena os processos RT pelo tempo que falta até falharem o deadline
  - O escalonamento pode estar feito estaticamente (se os processos estão fixos)

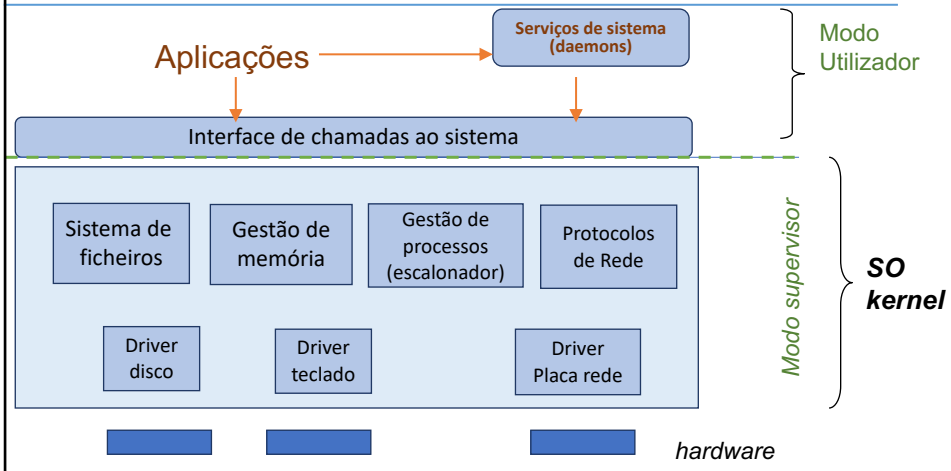
Implementação do SO

- Monolítico
  - Camadas, módulos, *device drivers*
  - Servidores (*daemons*)
- Outra abordagem: Microkernels

Sistema monolítico

- O SO é como um único programa que executa em modo supervisor:
  - Um só espaço de endereçamento
  - Eventualmente *multithreaded*
- No entanto o software é estruturado e modular:
  - Vários níveis (serviços user level; interface; implementação de serviços/gestores; device-drivers)
  - Modulares (gestor memória; sistema de ficheiros; gestor de processos, escalonador; device-drivers)
  - Possivelmente com ligação dinâmica de módulos de kernel para reconfiguração

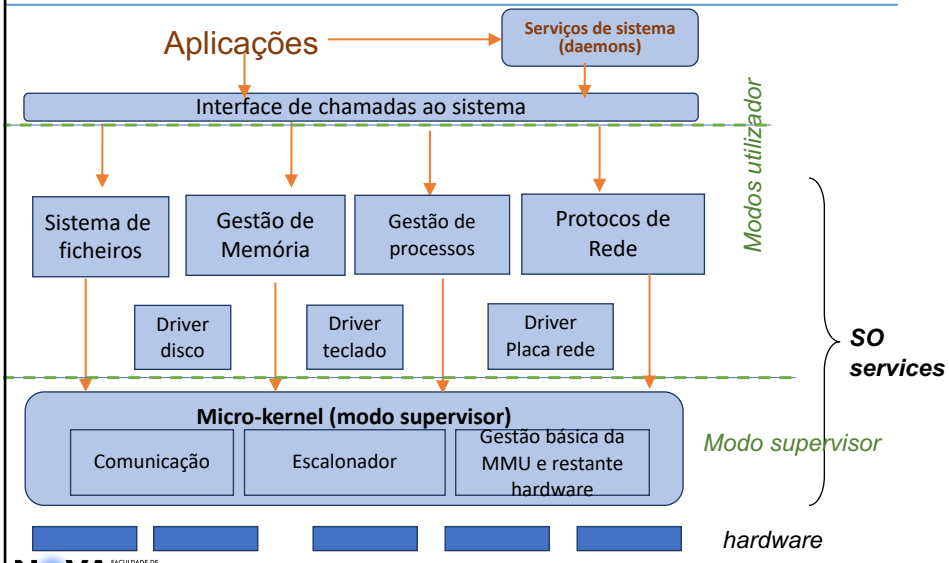
Organização do software



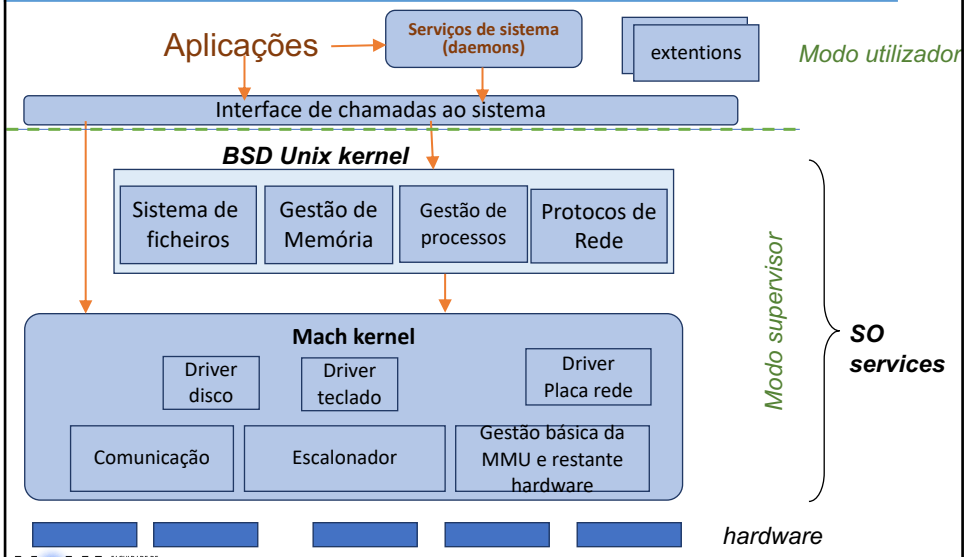
Micro-kernels

- Os vários níveis ou módulos são programas distintos:
  - Vários espaços de endereçamento; comunicam por troca de mensagens (cliente/servidor)
  - Apenas um nível básico necessita de executar em modo supervisor: tratamento de interrupções; gestão de páginas; troca de mensagens do sistema; escalonamento.
    - depende de SO para SO
  - Maior proteção e flexibilidade interna ao SO
    - Módulos/drivers isolados entre si contra crashs
    - pode reiniciar qualquer módulo com problemas ou substituí-lo
  - Maior *overhead*

Organização do software



Exemplo MacOS/Darwin



19

Mais virtualização...

- Outras formas de isolamento e virtualização?
- Para além dos processos e threads
  - com utilizadores e suas proteções
- Quando queremos ambientes para grupos de processos
  - Controlando o uso de memória, IO/disco/rede, ...
  - Diferentes serviços, bibliotecas e configurações do SO
- Ou mesmo diferentes OS ...
- Ou, quando queremos consolidar num computador vários sistemas
  - flexibilidade, reconfiguração e melhor uso de recursos

20

20

Máquinas virtuais

- Máquina virtual pode ser um qualquer sistema que oferece uma "máquina" (API, recursos) para execução de um outro sistema
- As linguagens programação oferecem uma VM aos programas/programadores independente do SO e hardware
- Os SO oferecem uma VM aos Processos independente do hardware
  - abstrai o hardware e oferece abstrações mais alto nível
- Mas no limite, uma MV pode oferecer a emulação completa duma arquitetura de computador
  - oferece a interface do hardware para que um SO execute sem notar diferença

Virtualização de hardware

- *Virtual Machine*
  - Oferece a interface de uma máquina "real"
    - Arquitetura real ou semelhante a uma real
    - Pode oferecer uma arquitetura diferente da real em que executa (JVM, emuladores de computadores antigos ou consolas de jogos, etc)
  - Capaz de executar um SO com os seus processos
    - Virtualiza o modo supervisor, periféricos, etc
    - Acedendo aos periféricos reais ou com emulação
- Sobre o hardware real (host) podemos executar várias instâncias de VM
  - Para obter vários SO num computador
  - Uma forma de conseguir multiprogramação
    - IBM CP/CMS (1968), VM/370 (1972)

Tipos de hipervisores

- Hipervisor é responsável pela gestão da virtualização entre VMs
- Type 1: Corre nativo (como um microkernel). Ex: VMware ESXI, Xen
- Type 2: Um processo no SO host. Ex: VMware Workstation, VirtualBox