Hardware e Software das TI

Operating Systems

POS_Conhecimento

Operating System

Software that runs "directly on the hardware", in ***kernel*** mode prividing:

1. RESROUCE ABSTRACTION

- Allows abstraction from the actual hardware used, **standardizing** the interface that the user ineracts with.
- The **VIRTUAL MACHINE** hides the details of the hardware

2. RESROUCE ABSTRACTION

- Manages the hardware resources (CPU, memory, disks, etc)
- Each program (in reality each process) is given a bit of CPU time, and access to the printer, memory, disks, network cards, etc.

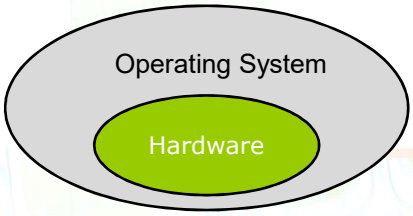
Introduction

PURPOSE OF THE OPERATING SYSTEM

- Manage resourced
- Create a standard virtual machine

DIFFERNCES BETWEEN THE VIRTUAL MACHINE AND THE REAL ONE

- Input / Output
- Main Memory
- File System
- Protection and error management
- Interaction with different programs
- Program control



Operating Systems

Babagge and the analitical machine (1871)

1st generation : 1945 – 1955

Vaccum tubes

Plug boards

2nd gereration: 1955 – 1965

Transistores

Batch Systems

3rd generation: 1965 – 1980

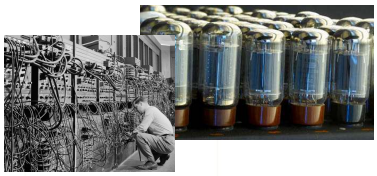
Integrated Circuits

Multiprogramming

4th generation: 1980 – present

PCs

And now…



History and types of O.S.

Control Monitors

- Resonsibe for system boot, loading programs, using I/O routines, and sometimes have na command interpreter.

Batch Sytems (or Job-shop Systems)

- Automatically load the next "JOB" in the queue, and have an elementary control over memory and process execution

Multiprogrammed Systems (or Cuncurrent Systems)

- Allow various concurrent processes.

Interactive Systems

- Allow man-machine interaction during program executions.

Vitual Memory Systems

- Flexible memory management

Distributed Systems

- Manage several machines (or several processing elements)

1st generation : direct input of code

Only one process at a time

- Load the program code into the computer, together with the data that will be processed
- Execute the program
- Output the results (to a printer, to tape, to punched cards…)

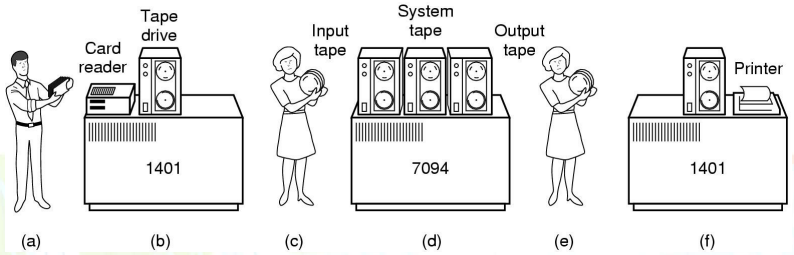
A lot of computer time is wasted:

- Slow loading of programs and data
- Slow output of results
- Very expensive computers makes the waste even worse !

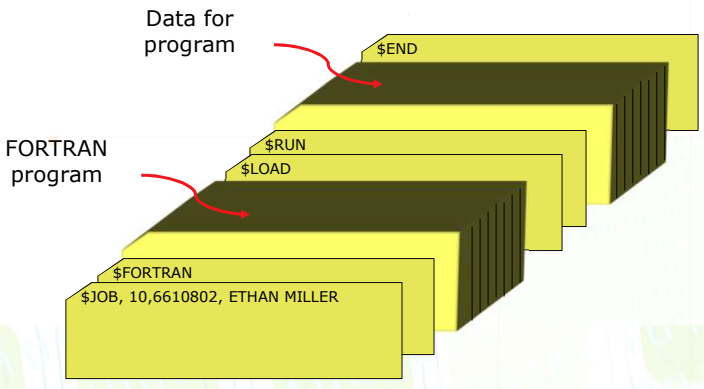
2nd generation: batch systems

Dedicated and more efficient I/O devices

- Read cards (with code and date) and transfer to tape on the 1401
- Place tape on the 7094 for fast reading and transfer to the computer
- Execute the program and write the results to another tape on the 7094
- Place the tape on the 1401 to read the outpt data and print it



"Job" structure for a 2nd generation computer process



Spooling

Initially bath systems used tapes
Mais tarde passaram a usar discos

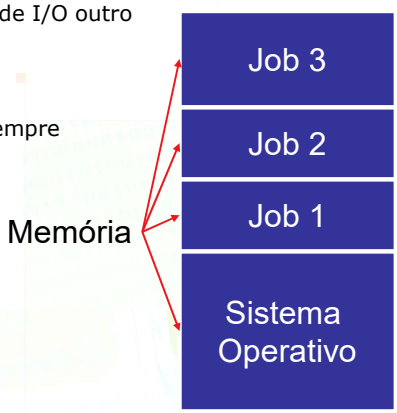
Spooling (*Simultaneous Peripheral Operation Online*)→ colocação de dados numa área de trabalho temporária onde outro programa possa aceder para processar no futuro.
Esta técnica consiste na colocação de *jobs* numa área da memória ou de um disco (buffer).

Assim, o operator carrega os cartões para um disco ligado ao computador
Computador lê jobs do disco, executa-os e escreve resultado para disco

Apenas 1 job de cada vez

3rd generation: multiprogrammed systems

- Várias tarefas em memória (em zonas separadas)
- Enquanto uma tarefa espera pelo processo de I/O outro job pode usar o CPU
- Se existir memória suficiente, o CPU tem sempre tarefas para executar



Tipos de Sistemas operativos

TIPOS DE SISTEMAS OPERATIVOS

- Mono-utilizador (por ex. MS-DOS)
- Controlo de processos (na industria, por ex.RTOS)
- Interrogação de ficheiros (base de dados)
- Processamento de transações (bancos,com.seg.)
- Uso geral (por ex. UNIX, VAX-VMS, OS/400)

SISTEMA DE **USO GERAL**

- Sistema operativo que tenta servir para tudo e a todos.
- Características:
  - Suporte de BATCH
  - CONCORRENTE
  - MULTI-ACESSO
  - INTERACTIVO

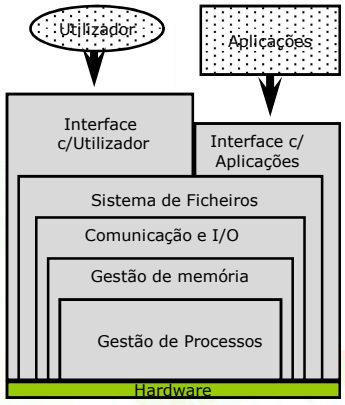
Arquitectura

Interface para o utilizador

- É necessário um **interpretador de comandos**.
- O interpretador pode ter **interface gráfica**

Interface para as aplicações

- É necessário uma **biblioteca de funções** do sistema operativo



Principais problemas

Concorrência

- Decidir que programa deverá executar (SCHEDULING)
- Evitar "*starvation*" de processos
- Dar temos de resposta aceitáveis
- Sincronizar os programas que têm necessidade disso
- Um S.O. é um **sistema não determinístico** !

Partilha de recursos

- Partilha de CPU, memória, periféricos
- Evitar que um programa interfira nos outros
- Decidir que informação ter em memória central, e qual deverá ser passada para disco

Armazenamento a longo prazo

- Sistema de ficheiros em disco

Conceitos básicos

PROGRAMA

- Sequência de comandos sem actividade própria.

PROCESSO

- Em primeira aproximação é um programa a correr
- Pode-se chamar também tarefa
- Um processo pode envolver a execução de mais de um programa, inversamente, um determinado programa pode estar envolvido em mais de um processo
- É algo dinâmico, que existe num período limitado no tempo, enquanto um programa é algo estático que tem uma existência ilimitada

PROCESSADOR

- É um órgão material que executa uma acção definida numa instrução máquina.

Gestão de processos

Informação associada aos processos - **CONTEXTO**

- Registos internos do processador
  - Importância de um stack próprio
- Memória e recursos associados ao processo

- Outras informações
  - Tempo de CPU gasto
  - Estado do processo (espera por um recurso)

MINIX process table

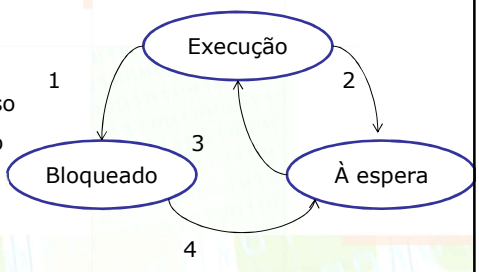
Process management	Memory management	File management
Registers	Pointer to text segment	Root directory
Program counter	Pointer to data segment	Working directory
Program status word	Pointer to stack segment	File descriptors
Stack pointer		User ID
Process state		Group ID
Priority		
Scheduling parameters		
Process ID		
Parent process		
Process group		
Signals		
Time when process started		
CPU time used		
Children's CPU time		
Time of next alarm		

Estados de um processo

1. Em **execução** -> Está a correr no CPU
2. À **espera** -> Pronto a executar, à espera de ter tempo de CPU
3. **Bloqueado** -> À espera de um recurso (impressora, discto, etc.)

Transições possíveis

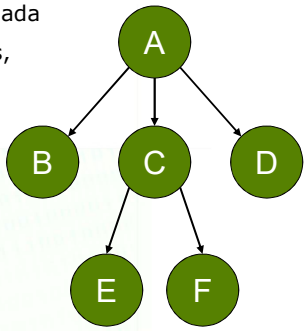
1. processo bloqueia à espera de input
2. SO (scheduler) escolhe outro processo
3. SO (scheduler) escolhe este processo
4. Input fica disponível



Criação de Processos

Processo: programa em execução

- Espaço em memória (memória que pode ser usada pelo programa), e Estado do processo (registos, program counter e stack pointer)
- Sistema operativo guarda esta informação de todos os processos (***process table***)



Um processo pode criar outro processo (**spawn**)

Processo A criou 3 processos filho: B,C e D

C criou 2 processos filho: E e F

Filhos normalmente herdam permissões dos pais

Conceitos básicos

COMUNICAÇÃO ENTRE PROCESSOS

- Os processos no interior do sistema não actuam isoladamente, devem cooperar: **troca de mensagens** e **memória partilhada**

EXCLUSÃO MÚTUA

- Os recursos podem ser classificados em:
  - Partilháveis
  - Não partilháveis
- Para os recursos não partilháveis, quando um processo o usa, é necessário excluir os outros

SINCRONIZAÇÃO

- A velocidade de um processo em relação a outro é imprevisível, visto que depende das interrupções e do tempo de processador que o sistema operativo atribui a cada processo. A certa altura é necessário que haja sincronização.

Threads

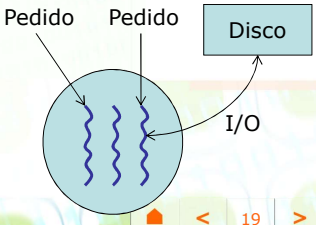
Também chamados de **processos leves**

Mecanismo para criar fluxos de execução, **partilhando um contexto comum**

Multi-threading → existem num processo vários threads

Exemplo de um processo de servidor de ficheiros:

- Recebe pedidos para ler ou escrever ficheiros e devolve o ficheiro ou aceita a edição
- Quando chega pedido é enviado para um thread
- Se esse thread ficar bloqueado à espera de informação do disco, outras threads estão disponíveis



Gestão de processos

Atribuição de "**time-slices**" aos processos

Existem vários critérios de atribuição de recursos

Execução paralela é mais eficiente (**1+1<2**)

- Cada processo pode demorar mais tempo
- O **desempenho global é melhor**
- Existe uma ocupação de recursos que nunca é perfeita

Time sharing simples

SO reparte o tempo de CPU pelos vários programas prontos a executar
- cria a ilusão que o computador está permanentemente disponível

Gestão de processos

Métodos para interromper os processos

Métodos **cooperativos** - após lhe ser atribuído o CPU, nunca mais lhe é retirado

Métodos **preemptivos** - desafectação forçada, o processo deixa de estar activo

Níveis de privilégios diferentes para o sistema operativo e para os processos

CPUs de concepção moderna suportam pelo menos 2 níveis de privilégio:

user level **-** nível de execução da generalidade das aplicações:
não permite aceder ao kernel space
não permite executar certas instruções do CPU

kernel level **-** nível de execução do SO
- sem restrições

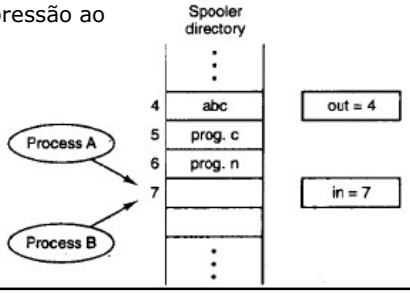
Partilha de recursos

Por vezes os processos podem partilhar um local de armazenamento comum (memória central ou ficheiro) onde podem escrever e ler

Exemplo: impressora e *print spooler*

1. Para imprimir ficheiro, processo envia nome para ***spooler directory***
2. Existem 2 variáveis partilhadas ***in*** e ***out***
   1. slot seguinte vazia
   2. próximo ficheiro a ser impresso
3. Processo A e B enviam ficheiro para impressão ao mesmo tempo

- Processo A lê in (7)
- Sinal de *clock* muda o processo para B
- Processo B lê in (7) e incrementa (8)
- Processo A é novamente chamado e escreve no slot 7



Secções críticas

Para garantir a coerência dos dados é necessário que os processos acedam ordenadamente aos recursos

SINCRONIZAÇÃO

Sempre que se testam ou se modificam estrutura de dados partilhadas

Deve ser feito dentro de uma **secção crítica:**

-parte do programa que acede a recursos partilhados

Garantir que 2 processos não estão na sua secção critica ao mesmo tempo

SINCRONIZAÇÃO

Sincronização é necessária para:
- Exclusão mútua
- Cooperação
- Acesso a variáveis

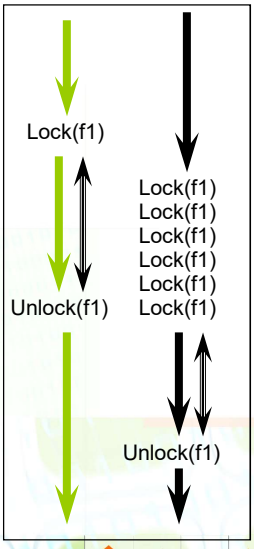
Sincronização com espera activa

while(not(flag1)) →sw

lock(flag1) →sw+hw

Sincronização com espera passiva

O processo suspende a sua execução



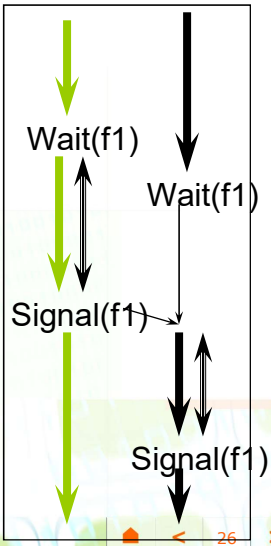
SEMÁFOROS

Servem para garantir exclusão mútua e sincronizar processos

Semáforo→ variável int para constar o nº de ***wakeups***

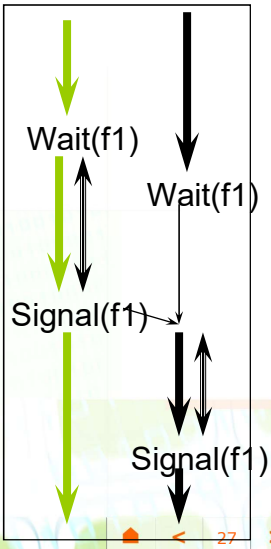
valor 0- nº wakeups guardados

valor *n*- *n* wakeups guardados



SEMÁFOROS

- Um processo só entra numa *ZONA CRÍTICA* se tiver "semáforo verde"
- Ao entrar na *ZONA CRÍTICA* o processo levanta um "semáforo vermelho"
- Os semáforos podem permitir mais do que um processo na *ZONA CRÍTICA* , ou seja podem ser "semáforos numerados"
- Um semáforo é constituído por uma variável de controlo, uma função de entrada (*wait*), e uma função de saída (*signal*)
- Em honra de Dijkstra, chama-se *p* e *v* a essas funções



SEMÁFOROS

FUNÇÃO WAIT

- É a função de entrada numa zona crítica
- Tem o aspecto *Wait(s),* em que s é um semáforo.
- Decrementa o valor de s se este for superior a zero. Se não, o processo fica bloqueado à espera que s torne o valor 1 ou superior.

FUNÇÃO SIGNAL

- Tem o aspecto *Signal(s)* em que s é um semáforo, isto é, uma variável inteira, positiva ou nula.
- Esta função incrementa o valor de s (eventualmente desbloqueando um processo que esteja à espera)

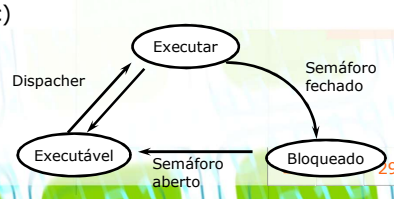
DEADLOCK (INTER BLOCAGEM)

- Quando vários processos competem entre si no uso de recursos é possível que ocorra a situação em que nenhum processo possa continuar, porque os recursos que um precisa estão reservados pelo outro e vice-versa.

DESPACHO

Estados possíveis de um processo

- Os processos são passados de um estado para outro por um programa dedicado, chamado DISPACHER (ou despacho)
- Os processos podem estar a EXECUTAR no CPU, podem estar EXECUTÁVEIS, mas em lista de espera para o CPU, ou BLOQUEADOS em semáforos
- Num sistema mais completo, um processo pode ainda estar suspenso em disco.
- Cada processo é representado por um *descritor* que contém toda a informação relevante (conteúdo dos registos do CPU, incluindo PC, SP, Acc, etc, nome do processo, prioridade, estado actual, etc)



Gestão de memória

Idealmente a memória seria

- grande, rápida e não volátil

Existe hierarquia na memória

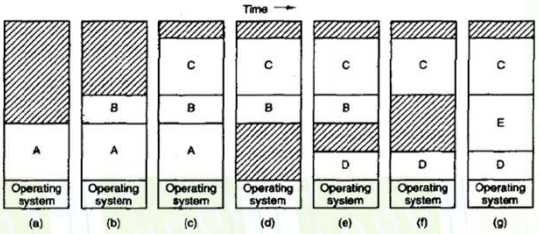
- memória rápida – cache - pouca
- velocidade média - memória principal - 1-4GB
- memória lenta – discos - >200 gigabytes

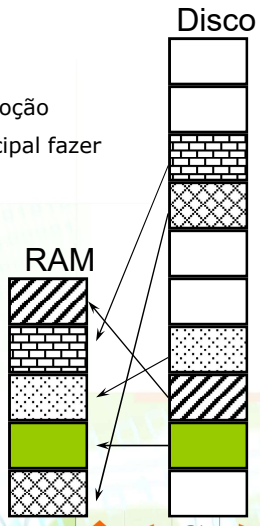
Esta hierarquia é gerida pelo gestor de memória

Gestor de memória

Faz parte do SO e é responsável pela:

1. Conhecimento das partes da memória estão em uso
2. Alocação da memória aos processos e sua posterior remoção
3. Gestão eficiente da memória - na falta de memória principal fazer *swapping* com o disco





Compactação de memória

Gestão de memória

Memória virtual

- A memória virtual é dividida em páginas
- **Page frames** -espaço correspondente na memória

- Ex: 32K de memória física e 64K de memória virtual
  16 virtual pages e 8 page frames

Relação entre endereços virtuais e físicos é dado por uma tabela