

Sistemas Operativos

LEI

2022/2023

fsm@.di.uminho.pt

.0

Numa palavra...

Se fosse necessário dizer de que trata esta UC

- Numa só palavra: **CONCORRÊNCIA**
- Em mais do que uma:
 - Concorrência, concorrência...
 - Eficiência, rapidez, segurança, etc.
 - Boa gestão de recursos perante determinada carga
 - Conhecimento do funcionamento de sistemas (Apps, SO, HW)
 - Aplicações concorrentes, naturalmente!

.2

Objectivo

- **Ajudar a perceber como funcionam os computadores**
 - Em termos físicos, o que é uma aplicação informática?
 - Que *recursos* necessita?
 - Como devem ser geridos esses recursos?
 - Como interage esta aplicação com outras?
 - Isto está lento. Que devo fazer?
 - O sistema bloqueou! Perdi tudo?
 - E muito mais...

.1

Programa

- Recapitulação de conceitos de **programação de sistemas**
- Gestão de processos, memória, ficheiros, periféricos
- Alguma **programação concorrente** (de baixo nível)
- E mãos na massa:
 - Aulas práticas em ambiente Linux
 - (quase) nada de janelas, nem ratos...
 - terminal com *Bash*, comandos, pipelines...
 - sem internet : basta consultar o manual
 - Programação de “baixo nível”: C, syscalls, libs ...



.3

.1

GSD
Grupo de Sistemas Distribuídos

Ambiente de trabalho: bash + ...

Sistemas Operativos - 2022/2023

4

.4

GSD
Grupo de Sistemas Distribuídos

Bibliografia recomendada

- [Operating Systems: Three Easy Pieces](#), Remzi & Andrea Arpaci-Dusseau, 2018.
- Silberschatz, Galvin and Gagne, *Operating System Concepts*, 10th Ed, John Wiley & Sons, 2021.
- **FSM 2004, Vou Fazer Exame de Sistemas Operativos!**

Sistemas Operativos - 2022/2023

5

.5

GSD
Grupo de Sistemas Distribuídos

Bibliografia Adicional

- Man !!!!!!!!
- R. Stevens, *Advanced Programming in the Unix Environment*, Addison Wesley, 1990.
- Beej Guides
- Google, Youtube, slashdot...

Sistemas Operativos - 2022/2023

6

.6

GSD
Grupo de Sistemas Distribuídos

Slides

- Baseados nos slides originais dos livros recomendados (em especial Silberschatz), caso seja necessário identificar o respectivo capítulo
- Disponíveis no Blackboard, mas em permanente revisão
- Servem apenas de “âncora” ao estudo

Não chegam para responder aos testes!

Faltam os porquês, e.g. porquê usar “isto” com esta “carga”?

Sistemas Operativos - 2022/2023

7

.7

.2

Aulas práticas

- Cada aula prática tem um “guião” muito detalhado.
 - Normalmente só duram uma semana
 - Fazem sempre falta para a aula seguinte (+ trabalho prático) →
 - Sempre que resolver uma alínea, deve parar e perguntar:
O que é que EU aprendi com este exercício?
- Recomenda-se:
 - estudar o guião antes da respectiva aula
 - usar a aula para tirar dúvidas e não para colecionar resoluções; tente entender o raciocínio para lá chegar.
 - terminar em casa todas as alíneas não resolvidas durante a aula; se necessário, peça ajuda por mail



.8

Avaliação

- Trabalho prático em grupo de 3, + um teste
 - Não há nota mínima no TP 
 - Não se “descongela” nota TP do ano anterior, salvo exceções indicadas no RAUM
 - Nota **mínima** de 8 no teste ou exame(s)

$$\text{Classificação} = (\text{TP} + \text{teste_ou_exame}) / 2$$

.9

Avaliação – datas previstas

- 13 Maio, Submissão do TP pelo Blackboard.
Coincide com a semana do Enterro da Gata!!!
- 15 a 19 de Maio -- Discussão presencial do trabalho prático
- 29 de Maio -- Teste escrito
- 19 de Junho -- Exame de recurso

.10

Avaliação

- Prova escrita (teste ou exame) individual e sem consulta.
- É preciso responder **ao problema proposto**;
 - Valoriza-se a capacidade de raciocínio e a concepção de algoritmos (por oposição à utilização de “padrões” de soluções)
 - Quase tudo tem a ver com concorrência
 - As perguntas da parte teórica insistem sempre nos “porquês”, na justificação, demonstração ou prova.

.11

Programa

- Introdução (à *programação de sistemas*)
- Gestão de processos
- Noções de programação concorrente
- Gestão de memória
- Gestão de periféricos
- Gestão de ficheiros

Sistemas Operativos - 2022/2023

12

.12

Vamos começar pelo princípio...

Para que serve um computador?

- Para executar programas (aplicações)
- Que facilitam a vida aos utilizadores

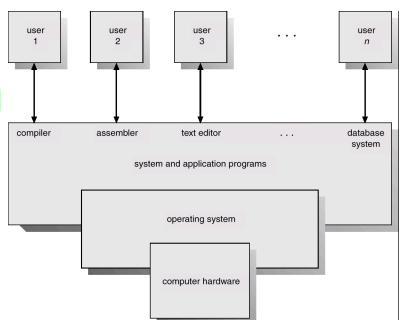
Sistemas Operativos - 2022/2023

13

.13

O que é um Sistema Operativo?

- Programa que actua como **intermediário** entre os utilizadores e o hardware



Sistemas Operativos - 2022/2023

14

.14

Portanto...

- O SO deve gerir o hardware e colocá-lo à disposição dos programas e utilizadores, de uma forma
 - conveniente,
 - protegida,
 - eficiente,
 - justa,
 - ...

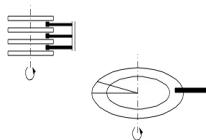
Sistemas Operativos - 2022/2023

15

.15

O Sistema Operativo pode ser visto como...

- Extensão da máquina
 - simula uma *máquina virtual* "acima" da máquina real: `open()`, `read()`, `write()`...
- Gestor de recursos



Sistemas Operativos - 2022/2023

16

.16

Objectivos (1)

- Conveniência
 - SO esconde os detalhes do hardware
 - e.g. [dimensão e organização da memória](#)
 - Simula máquina virtual com valor acrescentado
 - e.g. [cada processo executa numa "máquina"](#)
 - Fornece API mais fácil de usar do que o hardware
 - e.g. [ficheiros vs. blocos em disco](#)

Sistemas Operativos - 2022/2023

17

.17

Na prática...

- É o Sistema Operativo quem define a "**personalidade**" de um computador
- Como se comporta o mesmo computador (hardware) após ter arrancado
 - MSDOS?
 - Windows 95?
 - Windows 10?
 - Linux (Ubuntu, Kali...)?



Sistemas Operativos - 2022/2023

18

.18

Objectivos (2)

- Eficiência
 - SO controla a alocação de recursos
 - Se 3 programas usarem a impressora ao mesmo tempo → sai lixo?
 - Programa em ciclo infinito → [computador bloqueia](#)?
 - Processo corrompe a memória dos outros → [programas morrem](#)?
 - Multiplexação:
 - Tempo: cada processo usa o recurso à vez (impressora, CPU)
 - Espaço: recurso é partilhado simultaneamente por vários processos (memória central, disco)

Sistemas Operativos - 2022/2023

19

.19

Objectivos (3)

- Recapitulemos então os objectivos gerais de um SO
 - Conveniência
 - Eficiência
- Então, os nossos critérios de avaliação serão...
 -  Dá jeito?
 -  É eficiente ou aumenta a eficiência geral do sistema?
 -  Nem uma nem outra?

Sistemas Operativos - 2022/2023

20

.20

Tome nota:

- Este “filme” não é para decorar...
- É para perceber a evolução e os porquês
- Quando terminar, terá ficado a saber
 - os objectivos dos Sistemas Operativos
 - como estes foram sendo atingidos:
 - com muita massa cinzenta
 - e algum apoio do hardware!

Sistemas Operativos - 2022/2023

21

.21

No início era assim...

- Acesso livre ao computador
 - Utilizador podia fazer tudo, mas
 - Também tinha de fazer tudo...
- Eficiência era baixa
 - Elevado tempo de preparação
 - Tempo “desperdiçado” com debug

Sistemas Operativos - 2022/2023

22

.22

No início era assim...

 Exemplo: [HP 2114B \(1968\)](#)


Sistemas Operativos - 2022/2023

23

.23

- Comprava-se hardware sem software!
- Dava-se acesso livre ao computador
 - Utilizador podia fazer tudo
(i.e. interagir com o programa)
 - Mas também tinha de fazer tudo...

Sim, TUDO!!!!

Altair 8800 (intel 8080)



Sistemas Operativos - 2022/2023

24

.24

- A tradução manual é particularmente penosa e facilmente sujeita a erros
- É necessário construir a instrução a partir do “OP code”, flags, registos, endereços, modos de endereçamento...
- [8080 Instruction Set](#)

Inst	Encoding	Flags	Description
MOV D,S	01DDDS	-	Move register to register
MVI D,#	00DD110	db	Move immediate to register
LXI RP,#	00RP0001	lb hb	Load register pair immediate
LDA a	00111010	lb hb	Load A from memory
STA a	00110010	lb hb	Store A to memory
LHLD a	00101010	lb hb	Load H:L from memory
SHLD a	00100010	lb hb	Store H:L to memory
LDAX RP	00RP1010	*1	Load indirect through BC or DE
STAX RP	00RP0010	*1	Store indirect through BC or DE
XCHG	11101011	-	Exchange DE and HL content
ADD S	10000SS	ZSPCA	Add register to A
AND S	11000110	db	Add immediate to A

Sistemas Operativos - 2022/2023

26

.26

Acesso livre

- Utilizador é um faz-tudo (I):
 - Percebe o problema e idealiza a solução (algoritmo + dados)
 - Descreve o algoritmo em “alguma notação de alto nível”
 - Estuda o manual do CPU e memória, faz então a tradução para linguagem máquina (e decide que endereços vai usar)



```

1: S = 0
2: Ler N
3: Se N > 999 continuar no passo 6
4: S = S + N
5: Voltar ao passo 2
6: Mostrar S
  
```

```

$32, %rsp
$64, %eax
%eax, %edi
$0, -4(%rbp)
    _malloc
    _L_.str(%rip), %rdi
    8945 8313 03ff 1a25 0048 0073
    0002 0000 01ac 0000 0005 0007
    $4294967295, %ecx
    0048 105b 00eb f1f3 7502 4079
    %edx, %edi
    5d89 4cd8 0312 0000 0000 0000
    05400 40000 0000 07eb ff48 48c3 5d89 01d0 3bb6
    06420 8440 78ff 410a 448b 3cbe 2144 ebfb be0a
    05440 4000 0000 55e8 0001 8500 75c0 0fd9 03b6
    catq
    leaq
    movl
    xorl
  
```

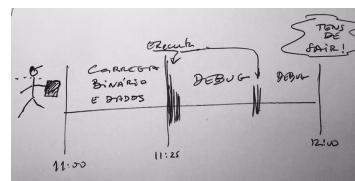
Sistemas Operativos - 2022/2023

25

.25

Acesso livre

- Utilizador é um faz-tudo (II):
 - Chegada a hora, carrega manualmente o programa e dados
 - Executa o programa
 - Se não está correcto, tem de descobrir sozinho os erros
 - Eficiência é muito baixa devido ao desperdício de tempo de CPU



Sistemas Operativos - 2022/2023

27

.27

- Carregamento manual demorado
- Debug muito demorado...
 - Erro no algoritmo?
 - Ao traduzir para assembly?
 - Ao traduzir para binário?
 - Ao carregar programa e dados?

Para aumentar a eficiência (I)

- Surgiu a ideia de reduzir a intervenção humana, fazendo a preparação de programas e dados *off-line* e acelerando o seu carregamento posterior.
 - Inventam-se periféricos de *input* e *output*, como a célebre Teletype. Apesar da baixa velocidade (10 chars por segundo... Seria muito melhor do que à mão.
 - E aparece um **loader*** residente
- (*) Software que entende o que está na fita e carrega para no local pretendido da RAM os bytes que vai lendo



Sistemas Operativos - 2022/2023

28

When apps were cards or tape...



Sistemas Operativos - 2022/2023

29

Card reader + line printer



Sistemas Operativos - 2022/2023

30

Para aumentar a eficiência (II)

- Automatizou-se uma parte do “procedimento”
 - Utilizador deixa de interagir com o seu programa, usa fita perfurada ou coloca os cartões num cesto e espera... horas
 - Operadores recolhem o cesto periodicamente e colocam programas e dados no leitor. O sistema executa os jobs e imprime os resultados, que são devolvidos a determinadas horas
- **Ganhou-se em eficiência, perdeu-se em conveniência**
 - Um job que antes demorava uma hora é agora executado em segundos
 - *Turnaround* time de horas: entrega às 9, recebe às 19

Sistemas Operativos - 2022/2023

31

.30

.31

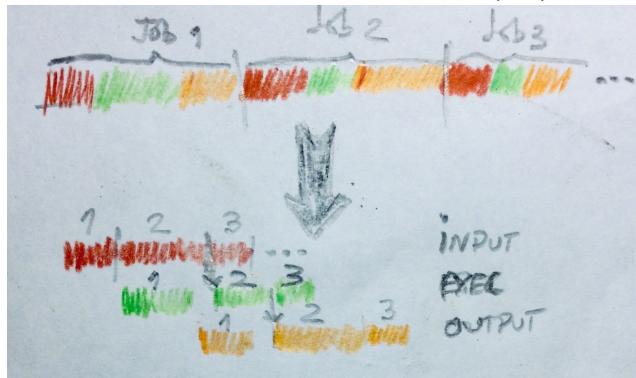
Melhor do que um operador...

- É ter um *programa* que:
 - Controla a operação do computador
 - Encadeia “jobs”, operador apenas carrega, descarrega e junta listagens aos respectivos cartões
- Será o início de uma Job Control Language (JCL) e de um interpretador de comandos ?

Embrião de um sistema operativo?

.32

Para aumentar a eficiência (IV)



.34

Para aumentar a eficiência (III) ...

Apesar de cada job “curto” demorar (dezenas de) segundos, o CPU está ocupado todo esse tempo, ora a executar o código ora em espera activa de IO.

E se... input, execução e output pudessem ser **realizados em paralelo**?

.33

Como gerir a execução concorrente?



.35

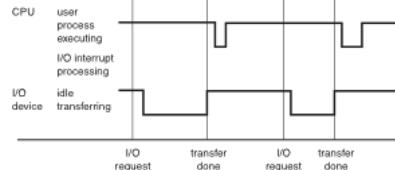
Mas havia o risco de...

- Perder eficiência devido a erros de programação
 - Ciclos infinitos
 - Espera por periféricos lentos
 - Erros na leitura ou escrita de periféricos
 - Programa do utilizador destruir o “programa de controle”

.36

Soluções (hardware)

- Interrupções



- Relógio de **Tempo Virtual**

- Instruções privilegiadas, 2 ou mais *modos de execução*
- Protecção de memória

.37

Exemplo: Polling IO

- Disk_IO()
 - Carrega o controlador de disco com parâmetros adequados (pista, sector, endereço de memória, direcção...)
 - While (NOT IO_done); /* do nothing in a **busy** way*/

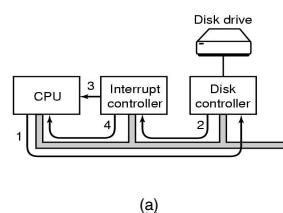
(Equivalente a:
 Já acabaste? Já acabaste? Já acabaste? Já acabaste? Já acabaste?
 Já acabaste? Já acabaste? Já acabaste? Já acabaste? Já acabaste?
 Já acabaste? Já acabaste? Já acabaste? Já acabaste? Já acabaste?
 ...)

- OK, regressa de disk_io()

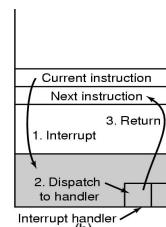
Resulta em **desperdício de tempo de CPU**

.38

Exemplo: Interrupt-driven IO



(a)



(a) OS inicia operação de IO e prepara-se para receber a interrupção; entretanto pode executar outras tarefas.

(b) No fim da operação de IO, o programa em execução é interrompido momentaneamente, SO trata o evento, e decide se é um processo que pediu IO que continua a executar ou troca para outro.

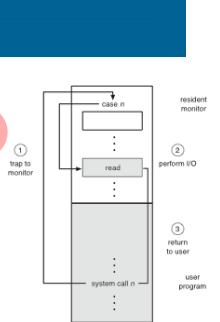
.39

Soluções (software)

- Chamadas ao Sistema
- Virtualização de periféricos

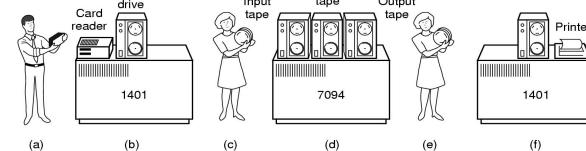
- Por exemplo o Leitor de cartões:
 - Programador pede para ler do periférico
 - SO devolve o conteúdo de um cartão que foi copiado para banda magnética ou lido anteriormente directamente para memória (SPOOL)

- Multiprogramação



.40

Primeiros sistemas de batch

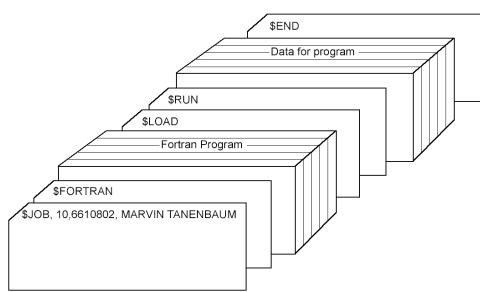


Processador auxiliar faz IO de periféricos lentos (virtuais)

- Carregar cartões no 1401, que os copia para banda magnética
- Colocar banda no 7094 e executar os programas
- Recolher banda com resultados e colocá-la no 1401, que os envia para a impressora

.41

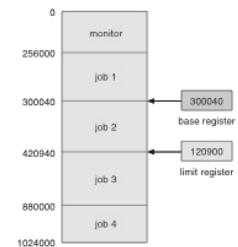
Exemplo de um “job”



.42

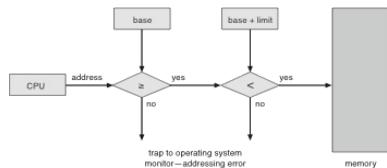
Multiprogramação

Vários jobs são carregados para memória central, e o tempo de CPU é repartido por eles.



.43

Protecção de memória



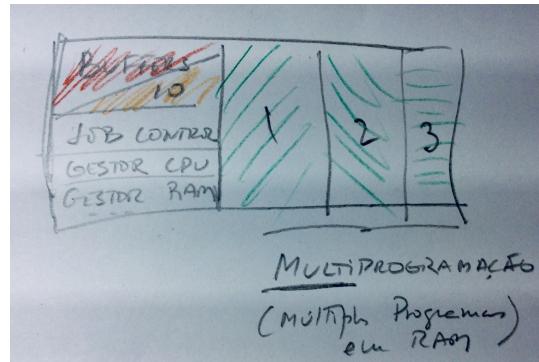
Note que estes testes têm de ser feitos sempre que há um acesso à memória...

2, 3 ou mesmo 4 vezes por instrução?

Sistemas Operativos - 2022/2023

44

Multiprogramação



Sistemas Operativos - 2022/2023

45

E a conveniência?



- Teve de esperar pelos sistemas de **Time-Sharing**
- Terminais (consolas) ligados ao computador central permitem que os utilizadores voltem a interagir directamente
- Sistema Operativo reparte o tempo de CPU pelos vários programas prontos a executar

Sistemas Operativos - 2022/2023

46

E desde aí?

- Com o computador pessoal volta tudo ao início...
 - Control Program for Microcomputers
 - Monoprogramação, baixa eficiência...
- Mas...
 - É muito conveniente para o utilizador
 - É barato, logo eficiência não é a prioridade

Sistemas Operativos - 2022/2023

47

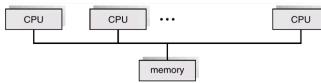
.46

.12

Multiprocessamento (1)

- **Vantagens**

- *throughput*
- economia
- *graceful degradation*
- ...



Exemplo: com 2 CPUs



A ideia é executar o dobro da carga no mesmo intervalo de tempo (i.e. maior *throughput*)

não é executar um programa mais depressa (i.e. baixar *tempo de resposta*). Para isso necessitaria de paralelizar a aplicação, dividi-la em vários processos

Multiprocessamento (2)

- **Arquitectura**

- Simétrico

- Qualquer CPU pode executar código do SO, mas
 - cuidado com *race conditions*, (e.g. tabela de blocos de memória livres)
 - hardware mais sofisticado (e.g disco interrompe todos os CPUs?)



- Assimétrico

- Periféricos associados a um só CPU, o que executa o SO
 - Não há *races*, mas os outros CPUs podem estar parados porque esse não “despacha” depressa,
 - nesse caso o *throughput* diminui

Sistemas Distribuídos (1)

- Nos anos 80 apareceram as redes locais para partilha de
 - recursos caros (e.g. impressoras) ou
 - inconvenientes de replicar (e.g. sistemas de ficheiros)
 - redirecionamento de IO

Exemplo: cat fich.txt | rsh print_server lpr

- **Questões**

- protocolos de comunicação, modelo cliente-servidor?
- como saber o estado de recursos remotos?

Sistemas Distribuídos (2)

- **Em breve**

- se passou dos *network aware OSs* para sistemas que estão vocacionados para o trabalho em rede
- as aplicações podem localizar e aceder recursos remotos de uma forma transparente



- E chegou-se à Web...

E ainda...

- SOs para *mainframes*:
 - IBM MVS, IBM VM/CMS.
 - desenvolvidos nos anos 60 e ainda em operação (z/VM)!
 - Actualmente a virtualização é (outra vez) **HOT TOPIC** (vmware, Xen...) + Docker...

Sistemas Operativos - 2022/2023

52

.52

E ainda...

- SO de Tempo Real
 - controlo de processos industriais, sistemas de vôo, automóveis, máquinas de lavar, etc.
 - SO normais não conseguem dar **garantias** de tempo de resposta.
- SOs para computadores “restritos”:
 - smartcards, PDAs, telemóveis, sensores...

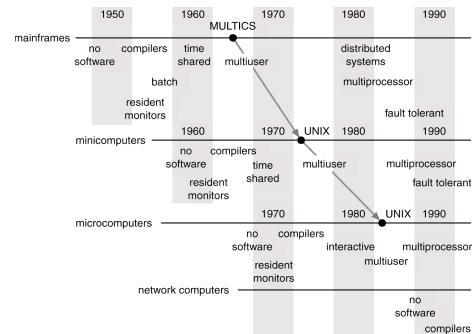


Sistemas Operativos - 2022/2023

53

.53

Evolução de conceitos de SO



Sistemas Operativos - 2022/2023

54

.54

Antes de continuar...



- Assegure-se que percebeu os conceitos anteriores, e que entendeu os problemas que as soluções indicadas procuram resolver...
- Por exemplo:
 - Sabe mesmo o que são e para que servem os 2 **modos de execução**?
 - Modo de execução é **hardware** ou **software**?
 - E multiprogramação? E multiprocessamento? E interrupção?
 - Para que servem as interrupções?
 - Para que servem as system calls? Qual a diferença em relação a uma função normal?
 - O que é o **tempo virtual**?

Sistemas Operativos - 2022/2023

55

.55

Arquitectura de Sistemas Operativos

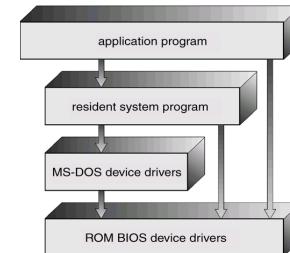
- Alguns exemplos
 - Sistemas monolíticos
 - Sistemas em camadas, hierárquicos
 - Modelo cliente-servidor
 - Máquinas virtuais
 - ...

Sistemas Operativos - 2022/2023

56

.56

MS-DOS Layer Structure

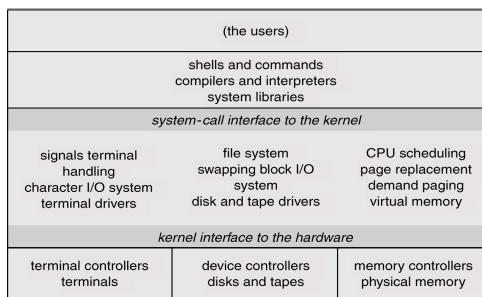


Sistemas Operativos - 2022/2023

57

.57

UNIX System Structure

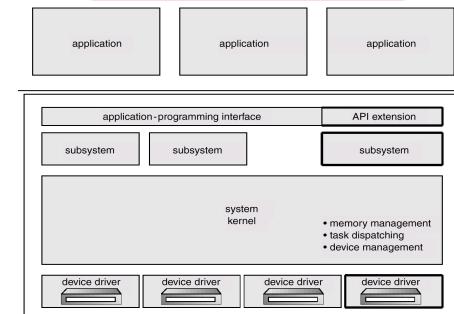


Sistemas Operativos - 2022/2023

58

.58

OS/2 Layer Structure

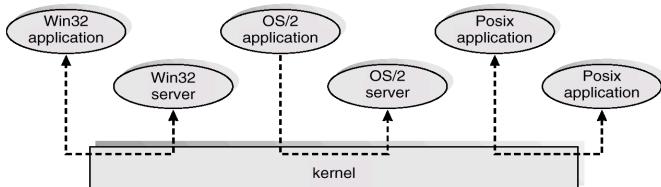


Sistemas Operativos - 2022/2023

59

.59

Windows NT (cliente-servidor)



E nas nossas aulas?

- O nosso SO é bastante modular
 - Módulo de gestão de processos
 - Módulo de gestão de memória
 - Módulo de gestão de periféricos
 - Módulo de gestão de ficheiros
- Mas há hierarquia / interdependência:
 - e.g. Memória virtual / memória real / disco / processos

Agora que já sabemos

- Para que serve um sistema operativo
- Quais os objectivos de um sistema operativo
- E começamos a saber:
 - como é um sistema operativo → **estrutura interna, algoritmos, ...**
 - e os porquês de ser assim
 - **que benefícios/objectivos se pretendem alcançar com determinadas estratégias**
 - **em que circunstâncias não se pode fazer melhor**

Convinha garantir que...

- Sabemos de facto
 - “Como é” um programa (e porquê?)
 - “Como é” um computador (e porquê?)
- Ou seja,
 - perceber as razões para o hardware e software de sistemas serem como são

O que é/como é um programa/processo?

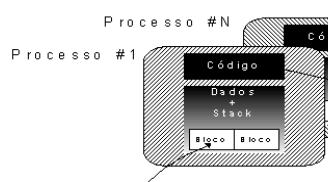
- Programa executável:

- Resultado da compilação, ligação, (re)colocação em memória
- Normalmente dependerá de módulos externos, libs

- Processo em execução:

- código já (re)colocado em memória central + dados +stack
- Estruturas de gestão:
 - Processo: contexto, recursos HW e SO em uso (registos, ficheiros abertos...)
 - Utilizador (uid, gid, account...)

The big picture



Dois programas a acederem simultaneamente ao mesmo ficheiro ou base de dados

O que é/como é um computador?

- CPU

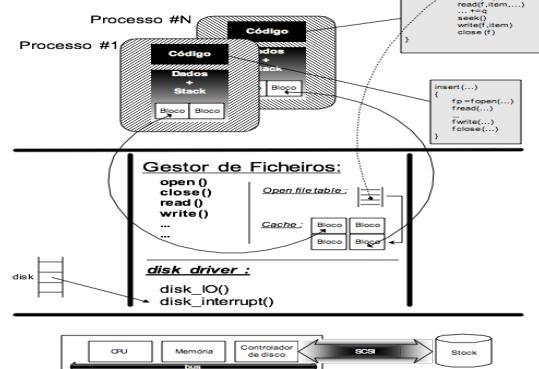
- Registos (PC, SP, BP, CS, DS...) → “contexto volátil”
- Instruções privilegiadas → só podem ser executadas em modo “protegido”; a forma de um programa do utilizador solicitar serviços ao SO é através das chamadas ao sistema (syscalls)

- Memória (mas o que é um endereço? E modos de endereçamento?)

- Periféricos + formas de dialogar com eles

- Interrupções (já agora, recordemos traps e exceções!)

Processo #1
Processo #N



The big picture revisited

- Assegure-se que percebeu

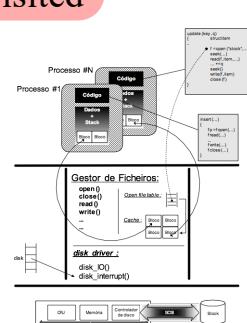
- Como surgem as “race conditions”

- entre processos
- dentro do SO

- Vantagens/desvantagens do uso de caches



- Note que estamos falar de caches por software, de cópias de dados em memória mas acessíveis em contextos diferentes



Programa

- Introdução

- Gestão de processos

- Noções de programação concorrente
- Gestão de memória
- Gestão de periféricos
- Gestão de ficheiros

Porquê criar vários processos?

- Porque dá jeito...

+ conveniência

- Estruturação dos programas

- Para não estar à espera (spooling, background...)

- Multiplas actividades / janelas

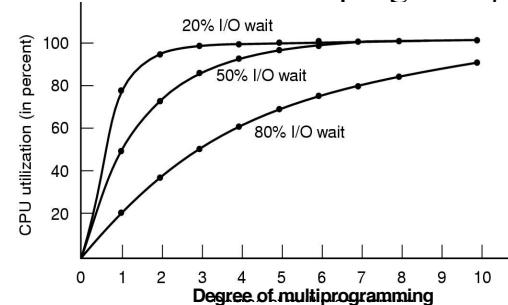
- Porque é melhor

+ eficiência

- Múltiplos CPUs

- Aumenta a utilização de recursos (e.g multiprogramação)

Benefícios da multiprogramação



Processos

- **Processo**: um programa em execução, tem actividade própria
- **Programa**: entidade *estática*, **Processo**: entidade *dinâmica*
- Duas invocações do mesmo programa resultam em dois processos diferentes (e.g. vários utilizadores a usarem cada um a sua shell, o vi, browser, etc.)

Processos

- O contexto de execução de um processo (i.e. o seu *estado*) compreende:
 - código
 - dados (variáveis globais, *heap*, *stack*)
 - estado do processador (registos)
 - ficheiros abertos,
 - tempo de CPU consumido, ...

Exemplo de informação sobre um processo

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		

Processos

- O SO deverá ser capaz de:
 - Criar, suspender e reiniciar a execução de processos
 - Suportar a comunicação entre processos
- O próprio SO tem muitos processos “do sistema”

Processos

- Para poderem executar os seus programas, os processos requerem tempo de CPU, memória, utilização de dispositivos...
 - Por outras palavras, os processos
- COMPETEM POR RECURSOS**
- E cabe ao sistema operativo fazer o escalonamento dos processos, i.e. atribuir os recursos pela ordem correspondente às políticas de escalonamento

Políticas de escalonamento

- Qual a melhor?
 - E a resposta é...
- Depende!
- De quem responde, utilizador ou administrador?
- É preciso definir **OBJECTIVOS**

Objectivos

- Conveniência
 - Justiça
 - Redução dos tempos de resposta
 - Previsibilidade
 - ...
- Eficiência
 - Débito (*throughput*), transacções por segundo, ...
 - Maximização da utilização de CPU e outros recursos
 - Favorecer processos “bem comportados”, etc.

Critérios de escalonamento

- IO-bound ou CPU-bound
- Interactivo ou não (batch, background)
- Urgência de resposta (e.g. tempo real)
- Comportamento recente (utilização de memória, CPU)
- Necessidade de periféricos especiais
- PAGOU para ir à frente dos outros...

Estados de um processo (i)



Processos em Unix

- Para criar um novo processo:
 - **fork**: cria um novo processo (a chamada ao sistema retorna “duas vezes”, uma para o pai e outra para o filho)
 - A partir daqui, ambos executam o mesmo programa
- Para executar outro programa
 - **exec**: substitui o programa do processo corrente por um novo programa
- Para terminar a execução
 - **exit**

Compare o **exec** com a invocação de uma função: são muito diferentes

fork/exec

```

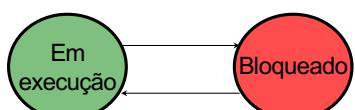
pid = fork()
if (pid == 0) {
    /* Sou o filho */
    exec( novo programa )
} else {
    /* Sou o pai
       A Identificação do meu filho é colocada na variavel pid
    */
}
  
```

fork'ing e exec'ing

- O padrão fork/exec é muito frequente (e.g. shell)
- Optimizações (a rever no capítulo de gestão de memória):
 - **copy on write**
 - Variante: **vfork**, não duplica o espaço de endereçamento; ambos os processos partilham o espaço de endereçamento e o pai é bloqueado até o filho terminar ou invocar o exec.

Estados de um processo (ii)

Podemos para já admitir que durante a sua “vida” os processos passam por 2 estados:



Sistemas Operativos - 2022/2023

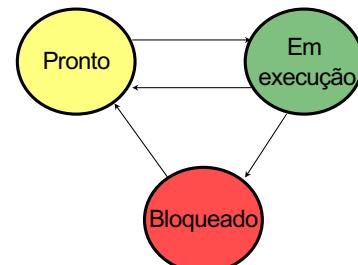
84

Estados de um processo (iii)

Na prática, há mais processos não bloqueados do que CPUs

Surge uma fila de espera com processos Prontos a executar

Processos em execução podem ser desafectados



Sistemas Operativos - 2022/2023

85

Estados de um processo (iv)

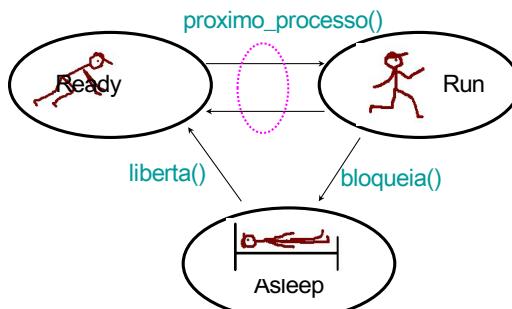
- Em execução
 - Foi-lhe atribuído o/um CPU, executa o programa correspondente
- Bloqueado
 - O processo está logicamente impedido de prosseguir, e.g. porque lhe falta um recurso ou espera por evento
 - Do ponto de vista do SO, é uma transição **VOLUNTÁRIA!**
- Pronto a executar, aguarda escalonamento



Sistemas Operativos - 2022/2023

86

Primitivas de despacho (i)



Sistemas Operativos - 2022/2023

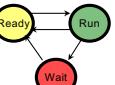
87

.86

.87

Primitivas de despacho (ii)

- Bloqueia(evento)
 - Coloca **processo corrente** na fila de processos **parados** à espera deste “evento”
 - Invoca `próximo_processo()`
- Liberta(evento) ou `liberta(processo,evento)`
 - Se o **outro** processo não está à espera de mais nenhum evento, então coloca-o na lista de processos **prontos a executar**
 - Nesta altura pode invocar ou não `próximo_processo()`



88

Sistemas Operativos - 2022/2023

.88

Primitivas de despacho (iii)

- `Proximo_processo()`
 - Seleciona um dos processos existentes na lista de processos prontos a executar, de acordo com a política de escalonamento
 - Executa a comutação de contexto
 - Salvaguarda contexto volátil do processo corrente
 - Carrega contexto do processo escolhido e regressa (executa o `return`)

 Como o Stack Pointer foi mudado,
 “regressa” para o **processo escolhido**!

Sistemas Operativos - 2022/2023

89

.89

Principais decisões

- Qual o próximo processo?
- Quando começa a executar?
- Durante quanto tempo?
- Por outras palavras,

Há desafectação forçada ou não?

Sistemas Operativos - 2022/2023

90

.90

Escalonamento de processos

- Quando, uma vez atribuído a um processo, o CPU nunca lhe é retirado então diz-se que o escalonamento é **cooperativo** (non-preemptive).
 - Exemplos: Windows 3.1, co-rotinas, `thread_yield()`
- Quando o CPU pode ser retirado a um processo ao fim do quantum ou porque surgiu outro de maior prioridade diz-se que o escalonamento é com **desafectação forçada** (preemptive)

Sistemas Operativos - 2022/2023

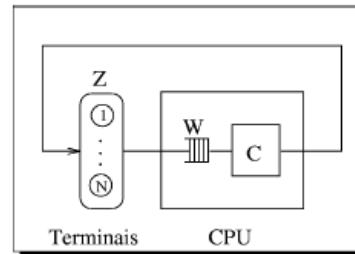
91

.91

Escalonamento de processos

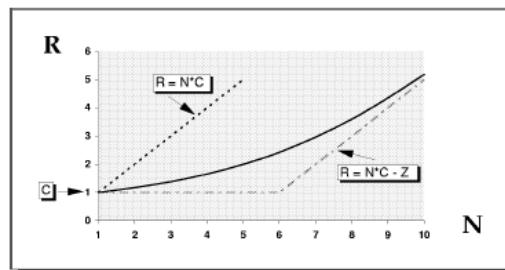
- Escalonamento **cooperativo** (non-preemptive).
 - “poor man’s approach to multitasking” ?
 - Sensível às variações de carga
- Escalonamento com **desafectação forçada**
 - Sistema “responde” melhor
 - Mas a comutação de contexto tem overhead

Modelo de sistema interactivo

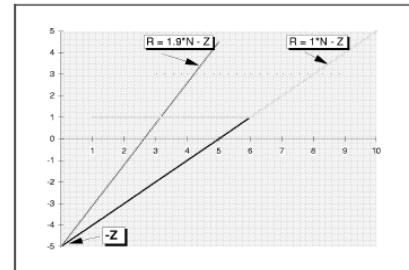


Z = Think time
 C = Service time
 W = Wait time
 N = Number of users

Tempo de Resposta (carga homogénea)



Tempo de Resposta (carga heterogénea)



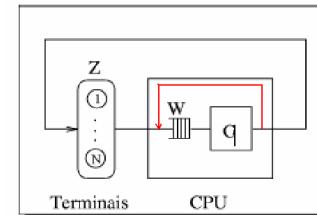
Assuma-se agora que uma em cada 10 interacções é muito longa, 10 vezes maior.
 Veja-se a degradação de tempos de resposta

Tempo de Resposta (carga heterogénea)

- Para evitar que as interações longas monopolizem o CPU e aumentem o tempo de resposta das restantes deve usar-se desafectação forçada.
- Neste caso deve atribuir-se um quantum (ou time slice) para permitir a troca rápida de processos:
 - Interacções curtas terminam dentro dessa fatia de tempo, logo não são afectadas pela política de desafectação.
 - Interacções longas executam durante um quantum e a seguir o processo correspondente regressa ao estado de **Pronto a Executar**, dando a vez a outros processos. Mais tarde ser-lhe-á atribuído nova fatia de tempo, e sucessivamente até a interacção terminar.

Duração da fatia de tempo

- Maioria das interacções deve “caber” num quantum
- $$R = W + C$$
- Se precisar de 2 passagens pelo CPU, $T_{Resposta}$ é quase o dobro!
- $$R = W + q + W + c'$$



Escalonamento de processos

- Escalonadores de longo-prazo (segundos, minutos) e de curto-prazo (milisegundos)
- Processo CPU-bound: processo que faz pouco I/O mas que requer muito processamento
- Processo I/O-bound: processo que está frequentemente à espera de I/O.

Escalonamento de processos

- Os processos prontos são seriados numa fila (*ready list*)
- A lista é uma lista ligada de apontadores para PCB's
- A lista poderá estar ordenada por prioridades de forma a dar um tratamento preferencial aos processos com maior prioridade

Escalonamento de processos

- Quando um processo é escalonado, é retirado da *ready list* e posto a executar
- O processo pode “perder” o CPU por várias razões:
 - Aparece um processo com maior prioridade
 - Pedido de I/O (passa ao estado de bloqueado)
 - O *quantum* expira (passa ao estado de pronto)

.100

Escalonamento de processos

- A decisão de escalar um processo pode ser tomada em diversas alturas:
 - Qdo um processo passa de a-executar a bloqueado
 - Qdo um processo passa de a-executar a pronto
 - Qdo se completa uma operação de I/O
 - Qdo um processo termina

.102

Escalonamento de processos

- Pretende-se maximizar a utilização do CPU tendo em atenção outros aspectos:
 - Tempo de resposta para aplicações interactivas
 - Utilização de dispositivos de I/O
 - Justiça na distribuição do tempo de CPU

.101

Escalonamento de processos

- Diferentes algoritmos de escalonamento visam objectivos diferentes:
 - Diminuir o tempo de resposta (reduzindo o tempo de espera para determinados processos)
 - Maximizar a utilização do CPU

.103

Escalonamento de processos

- Alguns algoritmos de escalonamento:
 - FCFS (First Come, First Served)
 - SJF (Shortest Job First)
 - SRTF (Shortest Remaining Time First)
 - Preemptive Priority Scheduling
 - RR (Round Robin)

Sistemas Operativos - 2022/2023

104

.104

First Come, First Served (FCFS)

- A *ready list* é uma fila FIFO
- Os processos são colocados no fim da fila e selecionado o da frente
- Método cooperativo
- Nada apropriado para ambientes interactivos

Sistemas Operativos - 2022/2023

105

.105

FCFS

- Tempo de espera com grandes flutuações dependendo da ordem de chegada e das características dos processos
- Sujeito ao “efeito de comboio”
- Uma vantagem óbvia do FCFS é sua simplicidade de implementação
- Parece haver vantagens em escalar os processos mais curtos à frente...

Sistemas Operativos - 2022/2023

106

.106

SJF (Shortest Job First)

- A ideia é escalar sempre o processo mais curto primeiro
- Possibilidades:
 - Desafectação forçada (SRTF) - interrompe o processo em execução se aparecer um mais curto
 - Cooperativo – aguardar pela terminação do processo em execução mesmo na presença de um processo recente mais curto

Sistemas Operativos - 2022/2023

107

.107

SJF

- Não se consegue adivinhar o tempo de processamento dos processos
- Apenas se podem fazer estimativas
- Usa uma combinação de tempos reais e suas estimativas para fazer futuras previsões.

.108

Preemptive Priority

- Problema: starvation
- Uma solução: envelhecimento – aumenta a prioridade dos processos pouco a pouco de forma a que inevitavelmente executem e terminem.

.110

Preemptive Priority

- Associa uma prioridade (geralmente um inteiro) a cada processo.
- A *ready queue* é uma fila seriada por prioridades.
- Escalona sempre o processo na frente da fila.
- Se aparece um processo com maior prioridade do que o que está a executar faz a troca dos processos

.109

RR (Round Robin)

- Dá a cada processo um intervalo de tempo fixo de CPU de cada vez
- Quando um processo esgota o seu quanto retira-o do CPU e volta a colocá-lo no fim da fila.
- Ignorando os overheads do escalonamento, cada um dos n processos CPU-bound terá $(1/n)$ do tempo disponível de CPU

.111

RR

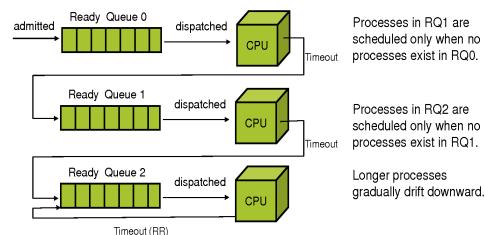
- Se o quantum for (muito) grande o RR tende a comportar-se como o FCFS
- Se o quantum for (muito) pequeno então o overhead de mudanças de contexto tende a dominar degradando os níveis de utilização de CPU
- Tem um tempo de resposta melhor que o SJF (o quantum “é” normalmente o SJ)

.112

Multilevel Feedback Queue Scheduling

- Another way to put a preference on short-lived processes
 - Penalize processes that have been running longer.

- Preemptive



.113

Níveis de escalonamento

- Uma vez que há inúmeros critérios de escalonamento (ie muitas variáveis a considerar para saber qual o “melhor” process, é habitual dividir a questão...
- 2 ou 3 níveis:
 - Nível 0 --- só despacha o que está em RAM
 - Nível 1 --- Decide que processos são multiprogramados
 - Nível 2 --- Não deixa criar processos nas horas de ponta

.114

- Introdução
- Gestão de processos
- Noções de programação concorrente
- Gestão de memória
- Gestão de periféricos
- Gestão de ficheiros

.115

Paralelismo versus concorrência

- Execução paralela => hardware
 - Vários computadores, eg. cluster
 - Multiprocessamento, eg. um processo em cada CPU
 - Hyper-threading / Dual core
 - CPU a executar instruções em paralelo com a operação de disco (que se manifesta através de uma **interrupção**, com prioridade superior à actividade no CPU)

Sistemas Operativos - 2022/2023

116

.116

Em geral...

- Seja num ambiente de paralelismo real ou simulado pelo SO
- Existem várias “actividades” em execução “paralela”
- Normalmente essas actividades **não são independentes**, há **interacção** entre elas
- Este facto que levanta algumas questões...

Sistemas Operativos - 2022/2023

117

.117

Papel do SO

- O sistema operativo tem a responsabilidade de
 - Fornecer **mecanismos** que permitam a criação e interacção entre processos
 - Gerir a execução concorrente (ou em paralelo), de acordo com as **políticas** definidas pelo administrador de sistemas

Sistemas Operativos - 2022/2023

118

.118

Cooperação e Competição

- Há normalmente 2 padrões de interacção entre processos:
 - Cooperam entre si para atingir um resultado comum
 - Processo inicia transferência do disco e aguarda passivamente que esta termine
 - O disco interrompe e a rotina de tratamento avisa que o processo já pode prosseguir
 - Competem por recursos (CPU, memória, impressora, etc.)
 - Há necessidade de forçar 1 ou mais processos a esperar até que o recurso pretendido fique disponível e lhes seja atribuído (seja a sua vez).

Sistemas Operativos - 2022/2023

119

.119

Sincronização

- Nos dois casos anteriores estamos perante uma questão de sincronização:
 - Cooperação (espera até que evento seja assinalado)
 - Competição (espera até que recurso esteja disponível)
- **Sincronizar** é fazer esperar, atrasar deliberadamente um processo até que determinado evento surja
 - Convém que a espera seja passiva.

Sistemas Operativos - 2022/2023

120

.120

Comunicação

- Para haver interacção tem de haver de **comunicação**:
 - Processos podem requisitar ao SO memória partilhada, podem escrever/ler ficheiros comuns, enviar/receber *mensagens* através de “canais”, pipelines, sockets, etc.
 - Threads do mesmo processo podem comunicar através de variáveis globais
- A comunicação pode ser tão simples como o assinalar a ocorrência de um evento (de sincronização), ou pode transportar dados.

Sistemas Operativos - 2022/2023

121

.121

Exemplos de concorrência

- Inspirados na vida real
 - Diálogo cliente/bar(wo)men
 - Actualização do saldo de uma conta bancária
 - Entrada num parque de estacionamento ou sala de cinema sem marcação de lugar
- São exemplos, respectivamente, de
 - Sincronização
 - Exclusão mútua (caso particular em que capacidade = 1)
 - Controlo de capacidade

Sistemas Operativos - 2022/2023

122

.122

Na realidade...

- Os três exemplos anteriores são casos de **Sincronização**, pois em todos poderá ser necessário esperar até que se verifique determinada condição:
 - Cliente: existir pelo menos um copo no balcão
 - O recurso estar livre, não haver outro processo a actualizar “esse” registo do ficheiro
 - Sala de cinema ou parque de estacionamento não estar totalmente cheios.
- Como veremos a seguir, conduzem a 3 padrões de soluções

Sistemas Operativos - 2022/2023

123

.123

Exemplo: num festival...

- Que algoritmo sugere para controlar a concorrência de largas dezenas de clientes ansiosos por retirarem um copo de cerveja do balcão?
- Que algoritmo sugere seja usado pela pessoa ou pessoas que estão a encher os copos e a colocá-los no balcão?

Sistemas Operativos - 2022/2023

124

.124

Não basta dizer...



- **Cliente/servidor ?**

- Já temos, não?
- Clientes pedem, barman/servidor responde?
- **Arranja-se uma fila**
- Onde? Só uma? Porquê? Como a implementa?
- **Lista ligada?**
- De quê? De copos, clientes ou barmen? Dos 3???

Sistemas Operativos - 2022/2023

125

.125

Onde está a espera?

- Uma lista é apenas uma estrutura de dados, quando muito poderia servir para comunicação entre barmen e clientes (balcão)
- E depois de inserir um copo nessa fila, como/quem espera por um cliente? Se a fila for de clientes, como esperam pelo seu copo?
- É preciso perceber a diferença entre **comunicação** e **sincronização**!

Sistemas Operativos - 2022/2023

126

.126

Vamos recapitular

- **Comunicação:** *escrita/leitura de dados* passados através de vários mecanismos (ficheiros, memória partilhada...)
- **Sincronização:** *coordenação de eventos* entre processos (usando semáforos, variáveis de condição, locks, etc.), designadamente para garantir uma ordem na ocorrência desses eventos -- só leio depois de tu escreveres;
- Há mecanismos que tratam dos dois aspectos, e.g. pipelines, mensagens, sockets...

Sistemas Operativos - 2022/2023

127

.127

Arquitectura da solução

- Que processos temos, quem são os componentes activos que executam algoritmos?
 - processos Cliente e processos Barman
- Como comunicam através do “balcão”?
 - variável/array/lista/ficheiro de copos?
- Como sincronizam?
 - Cliente espera por copo, barman pousa copo

Sistemas Operativos - 2022/2023

128

.128

Sincronização com variáveis partilhadas

CLIENTE

```
/* aguarda por copo cheio */
while (n_copos == 0);

n_copos = n_copos - 1;
tirar_copo_do_balcao();
...
```

Consegue perceber a ideia ?

Sistemas Operativos - 2022/2023

129

.129

Sincronização com variáveis partilhadas

CLIENTE

```
/* aguarda por copo cheio */
while (n_copos == 0);

n_copos = n_copos - 1;
tirar_copo_do_balcao();
...
```

BARMAN

```
/* aguarda vaga no balcão*/
while (n_copos == MAX);

n_copos = n_copos + 1;
pousar_copo_no_balcao();
...
```

Acha que este pseudo-código está correcto?

Sistemas Operativos - 2022/2023

130

.130

Infelizmente não está...

- Não garante que não haja copos partidos (?!)
- Nem clientes a pegarem no mesmo copo (?!?)
- E é muito ineficiente
 - esperas activas desperdiçam tempo de CPU
 - precisamos de primitivas capazes de *adormecer/acordar* processos

Sistemas Operativos - 2022/2023

131

.131

Mecanismos de sincronização

- Existem muitas propostas, sendo o seu estudo diferido para unidades curriculares como programação concorrente, sistemas distribuídos e computação paralela.
- Umas são genéricas (semáforos, mensagens), outras vocacionadas para casos particulares nomeadamente
 - exclusão mútua (e.g. mutexes, fcntl, Linux leases, métodos *synchronized*, etc.),
 - para serem usadas dentro do kernel do SO (sleep/wakeup, spin locks)
 - em ambientes multithreaded (wait/signal/notify)...

Sistemas Operativos - 2022/2023

132

.132

Semáforos

- Servem para resolver problemas de sincronização, exclusão mútua e controlo de capacidade
- Com apenas 3 operações*:
 - Inicialização :
 $s = \text{cria_semáforo}(\text{valor_inicial})$
 - $P(s)$ ou $\text{Down}(s)$
 - $V(s)$ ou $\text{Up}(s)$

* Na realidade há mais operações

Sistemas Operativos - 2022/2023

134

.134

Mecanismos de sincronização

- Em sistemas operativos explora-se a partilha entre processos de estado global tipicamente guardado em ficheiros e, por outro lado, o encapsulamento desse estado em processo(s) gestor(es).
- Nas aulas teóricas usam-se semáforos como introdução e treino mental de controlo de concorrência, enquanto que nas práticas se usam pipes e fifos, que combinam comunicação e sincronização no mesmo mecanismo.

Sistemas Operativos - 2022/2023

133

.133

Semáforos

- Imagine uma caixa com bolas e as operações:
 - P: Se há bola(s) na caixa, retiro uma e continuo, senão aguardo (passivamente) que alguém deposite uma
 - V: Devolvo a bola à caixa; se há alguém bloqueado à espera, acordo-o

(P = PÁRA e V = VAI; podem bloquear e libertar processos, respectivamente.)

Sistemas Operativos - 2022/2023

135

.135

Semáforos

```
P(s){
  s = s - 1
  if (s < 0)
    then bloqueia("S")
}
```

```
V(s){
  s = s + 1
  if (s ≤ 0)
    then liberta("S")
}
```

Quando $s < 0$, o seu valor absoluto $|s|$ conta o número de processos bloqueados no P()

Sistemas Operativos - 2022/2023

136

.136

Sincronização com semáforos

- Para cada evento de sincronização, é criado um semáforo com **valor inicial zero**
- Um processo espera *passivamente* pelo evento, e só avança depois do evento acontecer


```
P(s)      /* se caixa vazia, espera; senão evento já ocorreu */
```
- Outro processo assinala ocorrência do evento


```
V(s)      /* se ninguém à espera, deixa bola na caixa para indicar que o evento já ocorreu */
```

Sistemas Operativos - 2022/2023

138

.138

Semáforos

- Bloquear significa retirar o processo corrente do estado RUN e inseri-lo na fila "S"
- Libertar significa escolher um processo da fila "S" e inseri-lo na fila READY
- "S" contém os processos BLOQUEADOS no semáforo S
- Normalmente essa escolha é FIFO
- Mas pode não ser...**

Sistemas Operativos - 2022/2023

137

.137

CLIENTE

```
/* aguarda por copo cheio */
P(copo)
```

```
tirar_copo_do_balcao();
```

BARMAN

```
pousar_copo_no_balcao();
```

```
/* avisa que há +1 copo cheio */
V(copo)
....
```

Falta inicializar o semáforo copo a Zero

Sistemas Operativos - 2022/2023

139

.139

Capacidade

- Se a capacidade do balcão fosse de apenas 1 copo, Cliente e Barman executariam à vez. (Seria má estratégia... porquê?)
- Para aguardar por espaço no balcão, usa-se um semáforo inicializado à capacidade do recurso partilhado (e não a Zero)
- É um caso particular de sincronização: só bloqueia se o recurso estiver esgotado naquele instante

Sistemas Operativos - 2022/2023

140

.140

CLIENTE

```
/* aguarda por copo cheio */
P(copo)

tirar_copo_do_balcão();

/* avisa que há vaga */
V(espaço)
```

BARMAN

```
/* aguarda por vaga no balcão */
P(espaço)

pousar_copo_no_balcão();

/* avisa que há +1 copo cheio */
V(copo)
....
```

Falta inicializar o semáforo `copo` a Zero e `espaço` à capacidade do balcão

Sistemas Operativos - 2022/2023

141

.141

Exclusão mútua com semáforos

- Para cada região crítica, é criado um semáforo com valor inicial igual a **1 (um)**
- No início da região crítica


```
P(s)      /* só avança se região está livre */
```
- No fim da região crítica


```
V(s)      /* assinala que a região está livre */
```

Sistemas Operativos - 2022/2023

142

.142

“Receitas” com semáforos

- Sabendo que $\text{valor inicial} + \#V() \geq \#P()$ concluídos
 - Sincronização:
 $\text{valor inicial} = 0$
 - Exclusão mútua:
 $\text{valor inicial} = 1$
 - Capacidade:
 $\text{valor inicial} = N = \text{capacidade do recurso}$

Sistemas Operativos - 2022/2023

143

.143

Produtor/consumidor com semáforos

- Agora que resolvemos os aspectos de sincronização
- E já sabemos a “receita” da exclusão mútua
- É altura de reparar que o balcão é uma variável partilhada pelos vários processos (M barmen + N clientes)

=>Falta garantir **exclusão mútua** no acesso ao balcão!

.144

E não temos cliente/servidor?

- O que aqui temos são dois conjuntos de processos, uns a “produzem” dados e outros a “consumem” esses dados, de forma coordenada, eficiente e justa (ordem FIFO)
- Os algoritmos tratam apenas da passagem de dados entre produtores/barmen e consumidores/clientes/utentes. Mas poderão vir a ser usados numa arquitectura cliente/servidor!
- E antes de chegarmos a essa arquitectura, há que responder a:
 - *Que mal teria deixar os utentes/clientes servirem-se a eles próprios? Porque há um balcão e gente a servir?*

.146

Produtor(es)

```
P(espaço);
P(mutex);
Buf[p++ % N] = px;
V(mutex);
V(copo);
```

Consumidor(es)

```
P(copo)
P(mutex);
cx = Buf[c++ % N];
V(mutex);
V(espaço)
```

Falta inicializar os semáforos espaço a N, copo a ZERO, mutex a UM, e as variáveis p e c a ZERO.

.145

Self-scheduling vs Client/Server

- As chamadas ao sistema são executadas dentro do kernel mas em ambiente protegido e controlado
- Na arquitectura cliente/servidor há separação de processos:
 - o cliente nunca tem acesso directo ao estado do servidor nem AOS RECURSOS que este gere: isto implica permissões diferentes nos ficheiros e periféricos!
 - cliente faz o pedido e aguarda pela resposta: são precisos mecanismos de comunicação e sincronização, bem como de autenticação de clientes

.147

E não há...

- Sistemas operativos com arquitectura C/S?
 - Há! E desde os anos 80... tipicamente com um micro-kernel que gera escalonamento e diálogo com periféricos e um conjunto de processos “gestores” de recursos (e.g. o MACH de Carnegie Mellon University)
 - Curiosidade: dois dos investigadores principais do MACH mudaram-se em meados de 90 para a Microsoft e Apple, presumivelmente influenciando a evolução para Windows NT e MAC OSX...

.148

Programa

- Introdução
- Gestão de processos
- Noções de programação concorrente
- Gestão de memória
- Gestão de periféricos
- Gestão de ficheiros

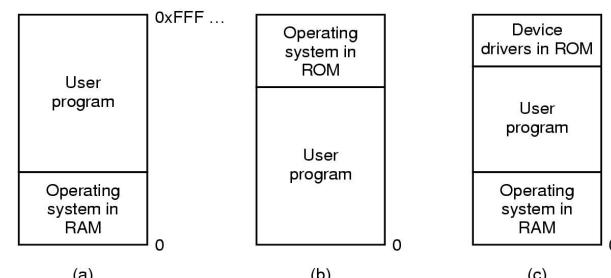
.149

Gestão de Memória

- Idealmente a memória seria:
 - grande
 - rápida
 - não volátil
- Na realidade, há uma hierarquia
 - Pouca memória muito rápida – cache(s)
 - Velocidade média, custo aceitável – RAM
 - Giga/terabytes de memória auxiliar – discos mag. /SSD

.150

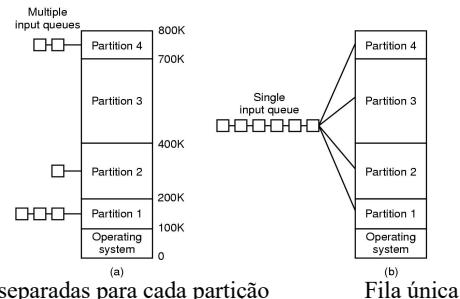
Monoprogramação



Três formas de organizar a memória com SO e apenas um processo

.151

Multiprogramação c/ partições de dimensão fixa (qq alteração implica reboot ?)



.152

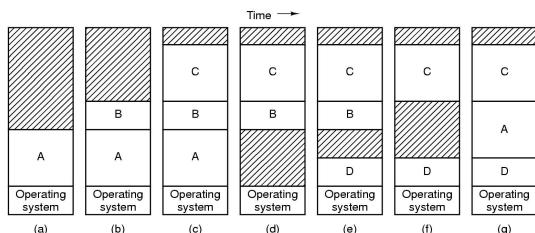
Recolocação e Protecção

- Para carregar um processo em qualquer partição livre...
 - O endereço de carregamento do programa não pode ser definido pelo compilador!
 - Isso é, o endereço de variáveis funções não pode ser *absoluto*
 - Tem de ser *recolocável*, para ser alterável no momento do carregamento
 - Para o compilador, *tudo começa no endereço 0!*
- CPUs passam a dispor de registos base e limite
 - Endereços saídos do CPU são adicionados à base para obter endereços físicos
 - Valores superiores ao registo limite são erros (=> processo é terminado)

.153

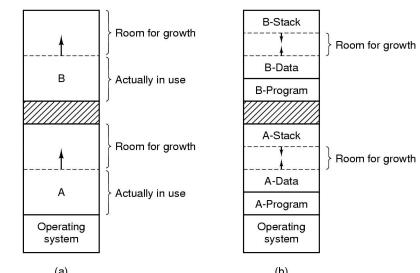
Swapping

A alocação de memória muda sempre que um processo é criado ou terminado (óbvio!), e também sempre que um processo é *swapped out* ou *swapped in* de disco



.154

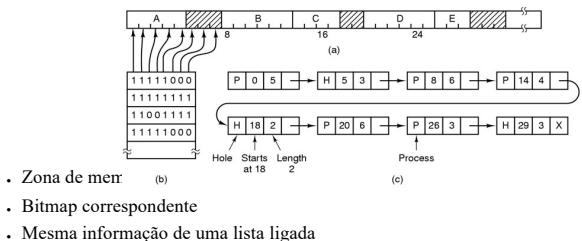
Swapping



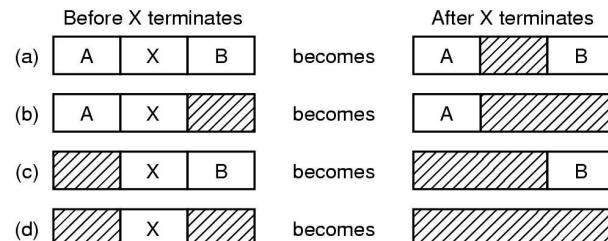
(a) Alocação para segmento de dados crescente
(b) Alocação para segmentos de dados e stack crescentes

.155

Gestão de memória com bitmaps



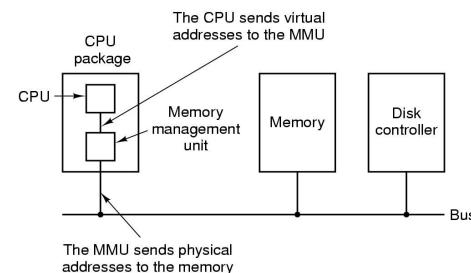
Gestão de memória listas ligadas



Em resumo, a GM baseada em partições...

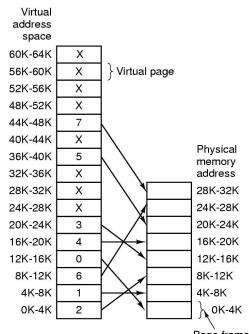
- Aumenta a eficiência à custa de multiprogramação mas...
 - **É inconveniente:**
 - Restringe a dimensão máxima dos processos
 - Alocação contígua dificulta a atribuição de endereços
 - Não permite protecção “fina” (e.g. read, write, exec)
 - **É ineficiente:**
 - Causa fragmentação e não permite partilha
 - Nem tira partido da dispersão de referências

Memória Virtual



Memória Virtual

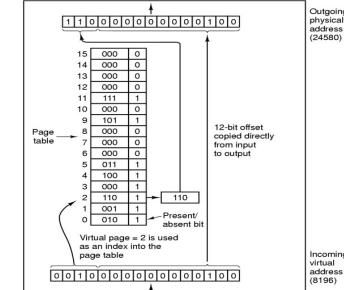
A relação entre endereços virtuais e físicos é dada por uma tabela



Sistemas Operativos - 2022/2023

160

Memória Virtual



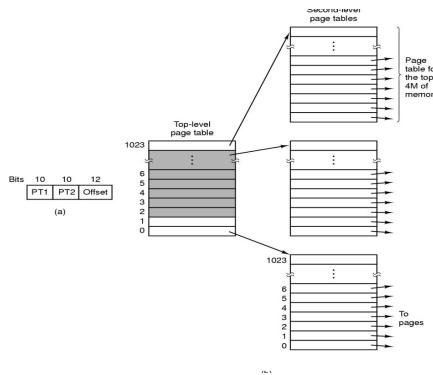
Operação da MMU com 16 páginas de 4 KB

Sistemas Operativos - 2022/2023

161

Memória Virtual

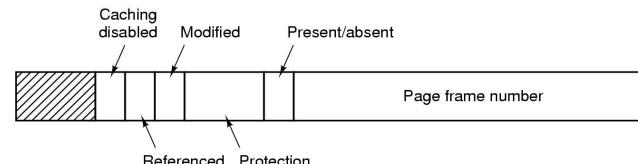
- Endereço de 32 bit c/ 3 campos:
 - PT1, PT2, Offset
- Tabelas de páginas de 2 níveis
 - Indexadas por PT1 e PT2
- Tabela de 2º nível tem endereço
 - base do frame respectivo
 - Ao qual se some o Offset



Sistemas Operativos - 2022/2023

162

Memória Virtual



Entrada típica da tabela de páginas

Sistemas Operativos - 2022/2023

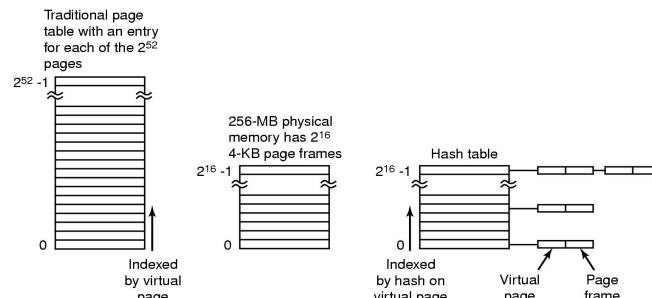
163

Memória Virtual

- TLBs – Translation Lookaside Buffers – para melhorar o desempenho

Valid	Virtual page	Modified	Protection	Page frame
1	140	1	RW	31
1	20	0	R X	38
1	130	1	RW	29
1	129	1	RW	62
1	19	0	R X	50
1	21	0	R X	45
1	860	1	RW	14
1	861	1	RW	75

Memória Virtual



- Comparação entre tabelas tradicionais e **tabelas invertidas**

Aspectos de Implementação

Tratamento da Page Fault:

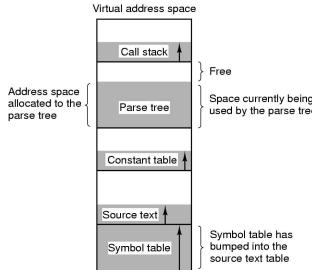
1. MMU interrompe o processador.
2. Kernel salva os registos e invoca função de tratamento
3. GMV determina a página necessária e a razão para interrupção
4. SO valida endereço e procura/cria page frame
 - Page in, zero fill, in transit...
 - Pode implicar rejeição de outra página

Aspectos de Implementação

O SO intervém na paginação em 4 situações:

1. Criação do processo
 - Determina o tamanho do programa
 - Cria a tabela de página
2. Execução do processo *ie, na comutação para o próximo processo*
 - Re-inicializa a MMU para o novo processo
 - Limpar a TLB
3. Page Fault
 - Determina o endereço virtual causador da *page fault*
 - Coloca a página em memória, se endereço for legal
4. Fim da execução do processo
 - Liberta a tabela de páginas e as páginas/frames

Segmentação



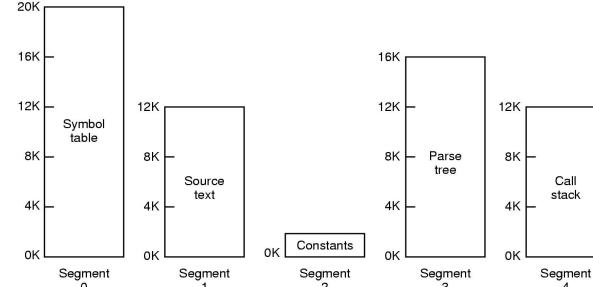
- Espaço de endereçamento único com tabelas crescentes
- Uma tabela pode sobrepor-se a outra

Sistemas Operativos - 2022/2023

168

.168

Segmentação



- Cada tabela pode crescer ou encolher independentemente

Sistemas Operativos - 2022/2023

169

.169

Segmentação vs Paginação

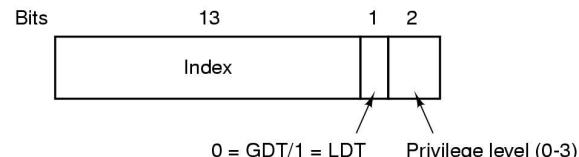
	Paginação	Segmentação
Transparente para o programador	Sim	Não
Número de espaços de endereçamento	1	Vários
O espaço de endereçamento pode ultrapassar o tamanho da memória física	Sim	Sim
O código e dados podem ser distintos e protegidos separadamente	Não	Sim
Tabelas de tamanho variável podem ser geridas facilmente	Não	Sim
A partilha de código é facilitada	Não	Sim

Sistemas Operativos - 2022/2023

170

.170

Segmentação com Paginação: Pentium

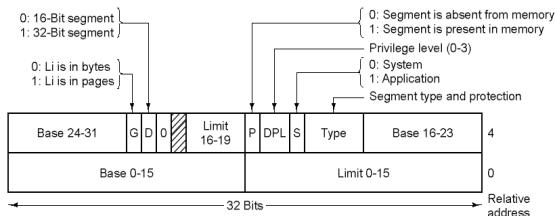

 Um **selector** no Pentium

Sistemas Operativos - 2022/2023

171

.171

Segmentação com Paginação: Pentium

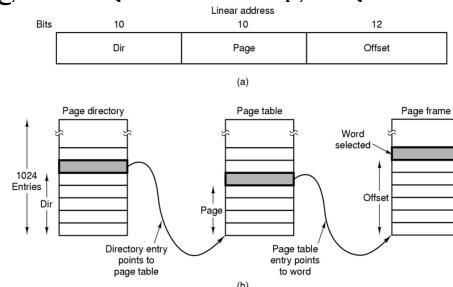


Descriptor de segmento de código

Sistemas Operativos - 2022/2023

172

Segmentação com Paginação: Pentium



Mapeamento de um endereço linear para o endereço físico

Sistemas Operativos - 2022/2023

173

Rejeição de páginas

- Um *page fault* leva:
 - A encontrar espaço para a nova página, a criar ou trazer de disco
 - A decidir que página em memória rejeitar, caso não haja espaço
- Uma página modificada tem que ser escrita (em disco)
 - Uma que não modificada pode ser imediatamente utilizada
- Convém não rejeitar uma página frequentemente usada
 - Pois provavelmente terá de ser carregada a seguir

Sistemas Operativos - 2022/2023

174

Rejeição de páginas

- Rejeitar a página que será usada mais tarde
 - Inexequível
- Aproximado por estimativa
 - Histórico de execuções anteriores do processo
 - Também isto é impraticável

Sistemas Operativos - 2022/2023

175

Rejeição de páginas NRU

- Cada página tem 1 bit de acesso e 1 de escrita
- As páginas são assim classificadas:
 1. Não acedida, não modificada
 2. Não acedida, modificada
 3. Acedida, não modificada
 4. Acedida, modificada

NRU remove a página com menor “ranking”

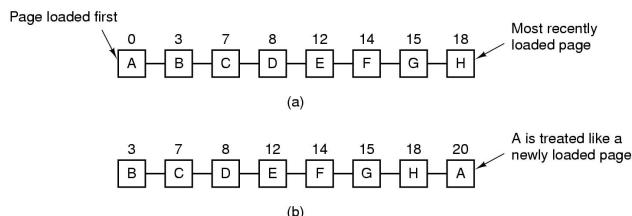
.176

Rejeição de páginas FIFO

- Mantém uma lista das páginas em memória
 - Segundo a ordem em que foram carregadas
- A página no topo da lista é rejeitada
- Desvantagem
 - A página há mais tempo em memória poderá ser a mais usada

.177

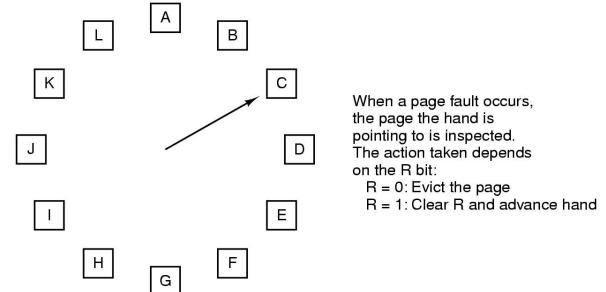
Segunda Oportunidade



- Ordem FIFO
- Se a página mais antiga tiver sido acedida, não é rejeitada
- É limpo o bit de acesso e é colocada no fim da fila

.178

Rejeição de páginas: Relógio



.179

Rejeição de páginas LRU

- Assume que as páginas usadas recentemente serão usadas em breve
 - Rejeitar a página não usada há mais tempo
- Tem de gerir uma (enorme?) lista de páginas
 - Ordenada pela mais recente
 - Tem de actualizar em todos os acessos à memória! (talvez não...)
- Uma alternativa seria manter um contador em cada entrada da tabela de páginas
 - Escolher a página com o menor valor
 - Periodicamente zerar o contador (de todas?)

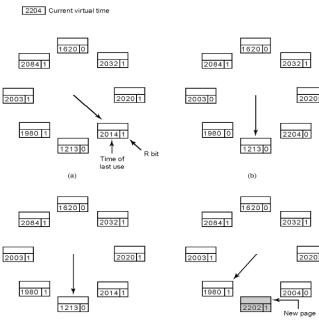
•180

Rejeição de páginas

Algoritmo	Características
Óptimo	Inexequível. Padrão para comparação.
NRU (não usado recentemente)	Aproximação grosseira.
FIFO	Leva à rejeição de páginas importantes.
Segunda Oportunidade	Melhoramento do FIFO.
Relógio	Solução realista.
LRU (menos recentemente usado)	Muito bom. Implementação exacta difícil.
NFU (menos frequentemente usado)	Aproximação grosseira do LRU.
Aging (envelhecimento)	Aproximação boa e eficiente do LRU.
Working set	Implementação ineficiente.
WSClock	Aproximação boa e eficiente.

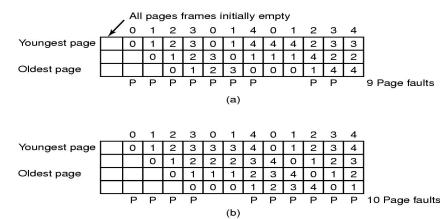
•182

WorkingSetClock



•181

Anomalia de Belady



- FIFO com 3 page frames
- FIFO com 4 page frames
- Os P's indicam ocorrência de page faults

•183

Gestão de Memória – recap

Como auto-avaliação, assegure-se que consegue acompanhar os vídeos seguintes:

<https://www.youtube.com/watch?v=qdkxXygc3rE>

<https://www.youtube.com/watch?v=qlH4-oHnBb8>

Sistemas Operativos - 2022/2023

184

.184

Sistemas Paginados

- Aspectos de Concepção de
 - Alocação local e global
 - Controlo de carga / thrashing
 - Tamanho das páginas

Sistemas Operativos - 2022/2023

185

.185

Alocação Local e Global

	Age
A0	10
A1	7
A2	5
A3	4
A4	6
A5	3
B0	9
B1	4
B2	6
B3	2
B4	5
B5	6
B6	12
C1	3
C2	5
C3	6

(a)

	Age
A0	10
A1	7
A2	5
A3	4
A4	6
A5	3
B0	9
B1	4
B2	6
B3	2
B4	5
B5	6
B6	12
C1	3
C2	5
C3	6

(b)

	Age
A0	10
A1	7
A2	5
A3	4
A4	6
A5	3
B0	9
B1	4
B2	6
B3	2
B4	5
B5	6
B6	12
C1	3
C2	5
C3	6

(c)

Config. original

Subst. local

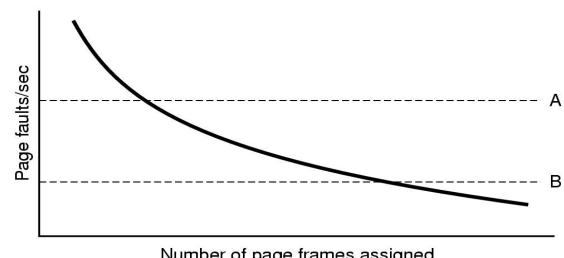
Subst. global

Sistemas Operativos - 2022/2023

186

.186

Alocação Local e Global



Sistemas Operativos - 2022/2023

187

.187

Controlo de carga

- Apesar de um bom desenho, pode ainda ocorrer **thrashing**
- Quando a Frequência de Page Faults é elevada e indica que:
 - Alguns processos precisam de mais memória central
 - Mas nenhum pode ceder parte da memória que tem
- A solução passa por reduzir o número de processos que competem por memória
 - Passando um ou mais processos para disco, libertando os frames que lhes estavam atribuídas. Mas quais???

Sistemas Operativos - 2022/2023

188

•188

Tamanho das páginas

Páginas pequenas

- Vantagens
 - Menos fragmentação interna
 - Melhor adequação a várias estruturas de dados e código
 - Menos partes de programas não usados em memória
- Desvantagens
 - Mais páginas, tabelas de páginas maiores

Sistemas Operativos - 2022/2023

189

•189

Tamanho das páginas

- Overhead estimado:

$$\text{overhead} = \frac{s \cdot e}{p} + \frac{p}{2}$$

Em que

 - s = tamanho médio dos processos em bytes
 - p = tamanho das páginas
 - e = entrada na tabela de páginas

$$\text{overhead} = \frac{s \cdot e}{p} + \frac{p}{2}$$

Diagrama que mostra o espaço da tabela de páginas (uma caixa com uma seta apontando para cima) e a fragmentação interna (uma caixa com uma seta apontando para baixo).

Valor óptimo quando

$$p = \sqrt{2se}$$

Sistemas Operativos - 2022/2023

190

•190

Tamanho das páginas

- Hardware pode impor valores fixos (e.g. 4KB em Intel) mas o SO pode *olhar para a memória* de outra forma, permitindo tamanhos maiores a certas aplicações
- Huge pages** de 2 MB... 2 GB ? Que não saem de RAM? **Porquê?**
- Quem se lembra do *sticky-bit* no início do Unix?

Sistemas Operativos - 2022/2023

191

•191

Programa

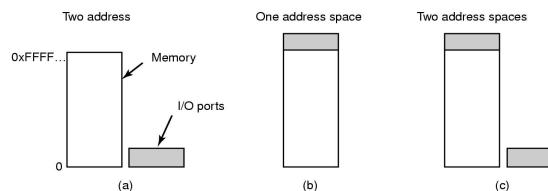
- Introdução
- Gestão de processos
- Noções de programação concorrente
- Gestão de memória
- Gestão de periféricos
- Gestão de ficheiros

Sistemas Operativos - 2022/2023

192

.192

I/O mapeado em memória



- Portas de E/S e memória separados
- E/S mapeado em memória
- Híbrido

Sistemas Operativos - 2022/2023

194

.194

Hardware de E/S

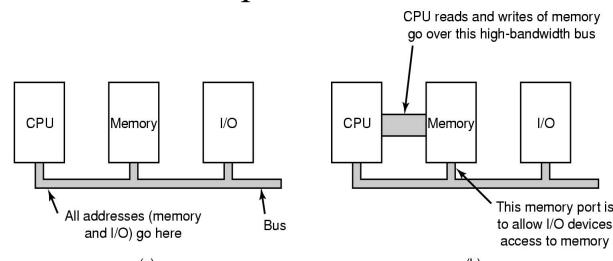
Device	Data rate
Keyboard	10 bytes/sec
Mouse	100 bytes/sec
56K modem	7 KB/sec
Telephone channel	8 KB/sec
Dual ISDN lines	16 KB/sec
Laser printer	100 KB/sec
Scanner	400 KB/sec
Classic Ethernet	1.25 MB/sec
USB (Universal Serial Bus)	1.5 MB/sec
Digital camcorder	4 MB/sec
IDE disk	5 MB/sec
40x CD-ROM	6 MB/sec
Fast Ethernet	12.5 MB/sec
ISA bus	16.7 MB/sec
IDE (ATA-2) disk	16.7 Mb/sec
FireWire (IEEE 1394)	50 MB/sec
XGA Monitor	60 MB/sec
SONET OC-12 network	78 MB/sec
SCSI Ultra 2 disk	80 MB/sec
Gigabit Ethernet	125 MB/sec
Ultrium tape	320 MB/sec
PCI bus	528 MB/sec
Sun Gigaplane XB backplane	20 GB/sec

Sistemas Operativos - 2022/2023

193

.193

E/S mapeado em memória

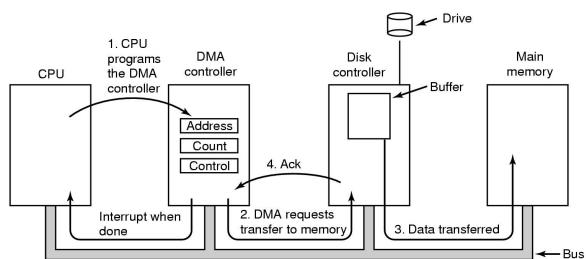


Sistemas Operativos - 2022/2023

195

.195

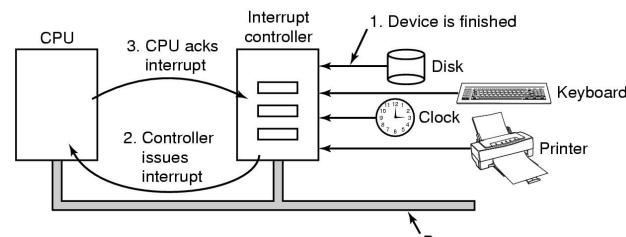
Memória de Acesso Directo (DMA)



Sistemas Operativos - 2022/2023

196

Interrupções



Sistemas Operativos - 2022/2023

197

Características do Software de E/S

- Independência de dispositivos
 - Programas podem aceder a qualquer dispositivo através de abstracções, sem que o tenham de especificar à priori.
- Uniformização de nomes
 - Nomes de ficheiros e dispositivos independentes da máquina
- Tratamento de erros
 - Tão perto do hardware quanto possível

Sistemas Operativos - 2022/2023

198

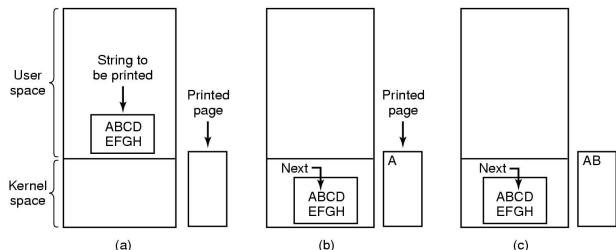
Aspectos do Software de E/S

- Transferências síncronas e assíncronas
 - Espera activa vs. interrupções
- Armazenamento temporário (buffering)
 - Armazenamento e cache em vários níveis
- Dispositivos partilhados vs. dedicados

Sistemas Operativos - 2022/2023

199

E/S programado



Fases na impressão de uma string

Sistemas Operativos - 2022/2023

200

.200

E/S programado

```
copy_from_user(buffer, p, count);           /* p is the kernel buffer */
for (i = 0; i < count; i++) {                /* loop on every character */
    while (*printer_status_reg != READY);    /* loop until ready */
    *printer_data_register = p[i];           /* output one character */
}
return_to_user();
```

Escrita de uma string para a impressora usando E/S programado

- Com espera activa?
- Sem Mutex?
- Acesso ao controlador (i.e. printer_data_register) ?

Sistemas Operativos - 2022/2023

201

.201

E/S com DMA

```
copy_from_user(buffer, p, count);      (a)
set_up_DMA_controller();
scheduler();
```

```
acknowledge_interrupt();             (b)
enable_interrupts();
unlock_user();
return_from_interrupt();
```

(a)

(b)

Operação de E/S com DMA

Sistemas Operativos - 2022/2023

202

.202

E/S por interrupções



```
copy_from_user(buffer, p, count);
enable_interrupts();
```



```
while (*printer_status_reg != READY) :
```

```
    *printer_data_register = p[0];
```

```
    scheduler();
```

```
if (count == 0) {
    unlock_user();
```



```
}
```

```
else {
```

```
    *printer_data_register = p[i];
    count = count - 1;
```

```

    i = i + 1;
```

```
}
```

```
acknowledge_interrupt();
```

```
return_from_interrupt();
```

(a)

(b)

(a) Inicia transferência e (b) Trata interrupção

Falta esperar pela impressora livre e que seja a minha vez, garantia de mutex neste “driver”, libertar o próximo, etc

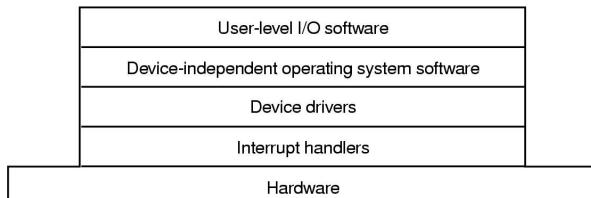
Sistemas Operativos - 2022/2023

203

.203

.51

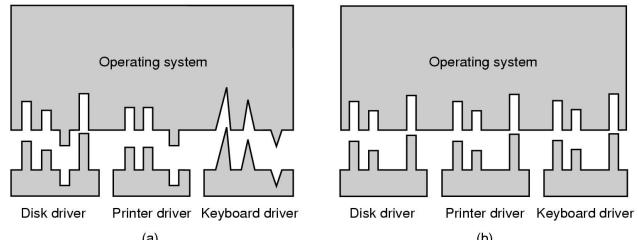
Níveis do software de E/S



Sistemas Operativos - 2022/2023

204

E/S independente do dispositivo



Sem interface standard

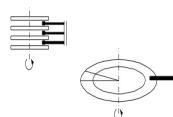
Com interface standard

Sistemas Operativos - 2022/2023

205

Discos

- O tempo necessário para aceder a um bloco é determinado por três factores:
 - Tempo de procura (posicionamento na pista)
 - Tempo de rotação do disco (posicionamento no sector)
 - Tempo de transferência
- O tempo de procura (seek) é dominante

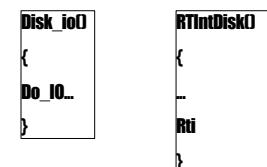


Sistemas Operativos - 2022/2023

206

Escalonamento de pedidos de transferência

- FIFO
- SSTF
- Elevator
- Scan circular


Consegue imaginar os algoritmos?

Como bloquear um processo até que chegue a vez do seu pedido?

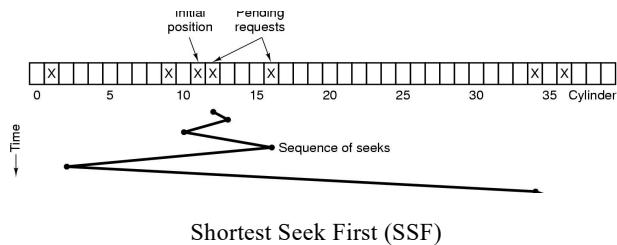
Sistemas Operativos - 2022/2023

207

.206

.207

Escalonamento de pedidos a disco

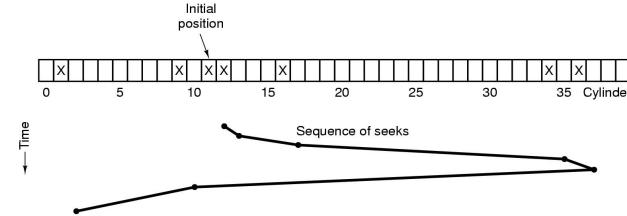


Sistemas Operativos - 2022/2023

208

.208

Escalonamento de pedidos a disco

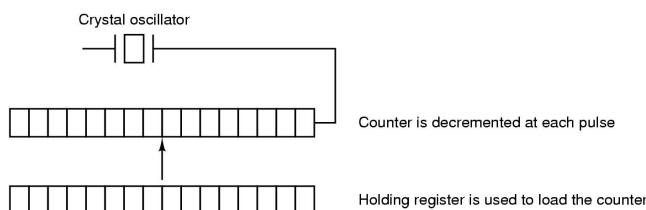


Sistemas Operativos - 2022/2023

209

.209

Relógios



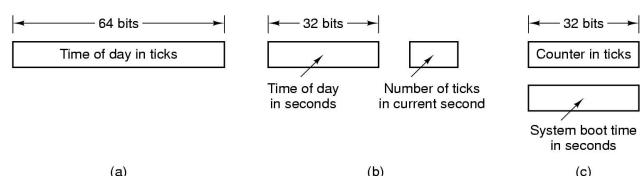
Um relógio programável

Sistemas Operativos - 2022/2023

210

.210

Relógios



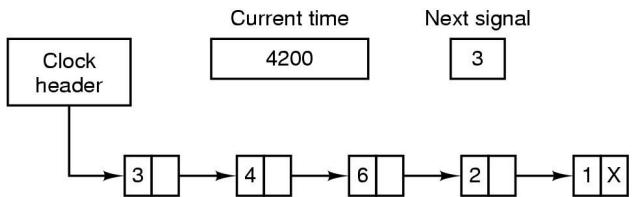
Três formas de manter a hora actual

Sistemas Operativos - 2022/2023

211

.211

Relógios



Simulação de vários contadores com um único relógio

Sistemas Operativos - 2022/2023

212

.212

Programa

- Introdução
- Gestão de processos
- Noções de programação concorrente
- Gestão de memória
- Gestão de periféricos
- Gestão de ficheiros

Sistemas Operativos - 2022/2023

213

.213

Gestão de Ficheiros

- Sistemas de ficheiros
 - Recapitulação de hw e sw de IO
 - . Discos, partições, disk IO, device drivers, concorrência, caches, etc.
 - Requisitos, objectivos, estudo de casos
 - RAID, Log structured File Systems
 - Noções de sistemas de ficheiros distribuídos

Sistemas Operativos - 2022/2023

214

.214

Sistemas de ficheiros: requisitos

- Persistência
- Grande escala (quantidade de ficheiros + dimensão elevada)
- Rapidez de acesso (Tempo de acesso a disco \gg TaccRAM)
- Concorrência
- Segurança
- ...

Sistemas Operativos - 2022/2023

215

.215

Objectivos (1)

- Armazenamento
 - Persistente (backup, undelete, RAID)
 - Eficiente
 - Espaço (=> aproveitar)
 - Dados (exemplos)
 - Alocação não contígua para eliminar fragmentação externa
 - Suporte para ficheiros “dispersos” (resultado de “hash”, por exemplo)
 - Metadados, eg. estruturas para representar blocos livres/ocupados: FAT, i-nodes, ...
 - Tempo: algoritmos de gestão e **recuperação** rápidos

Sistemas Operativos - 2022/2023

216

.216

Objectivos (2)

- Acesso
 - Escalável
 - Convenientes
 - estrutura interna visível (pelo kernel) ou só pelas aplicações?
 - Sequência de bytes vs. Ficheiros indexados
 - Seguro
 - controlo de acessos
 - auditoria
 - privacidade...

Sistemas Operativos - 2022/2023

217

.217

Objectivos (3)

- Acesso
 - Rápido (alguns exemplos de “bom-senso”)
 - Evitar dispersão de blocos pelo disco => cuidado na alocação, usando por exemplo
 - os “cylinder groups” do BSD, “file extents” do JFS e XFS
 - “hot file clustering” e desfragmentação “on-the-fly” do Mac OS X
 - Uso de caches (em disco e RAM) e delayed write => **CUIDADO!**
 - Directórios
 - Podem ter milhares de entradas (eg. e-mail!)
 - Procura sequencial? Binária? B-trees?

Sistemas Operativos - 2022/2023

218

.218

Objectivos (4)

- Acesso rápido
 - **Escalonamento de pedidos de transferência do disco** para minimizar movimentos do braço
 - A ideia é reduzir o tempo médio de acesso a disco
 - Como de costume, ao alterar a ordem de serviço, atrasa alguns pedidos em benefício de outros...
 - Recorde as várias estratégias:
 - FIFO, SSTF, SCAN, C-SCAN...
 - Consegue imaginar os algoritmos?

Sistemas Operativos - 2022/2023

219

.219

E se um disco tem uma avaria?



Sistemas Operativos - 2022/2023

220

.220

RAID

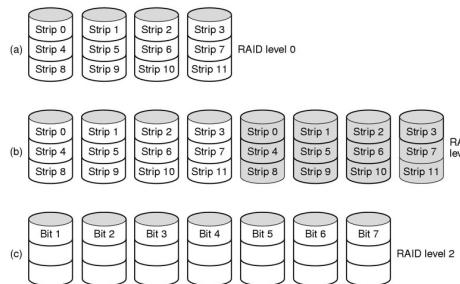
- Redundant Arrays of Inexpensive Disks
 - Pesquise no google por “Raid-1 Raid-5 primer”
- Objectivos:
 - Desempenho
 - Disponibilidade
 - Tolerância a faltas nos discos (depende do tipo de RAID)
 - Não resolve ficheiros apagados, virus, bugs, etc
 - Continua a precisar de BACKUPs!!

Sistemas Operativos - 2022/2023

221

.221

Sistemas RAID

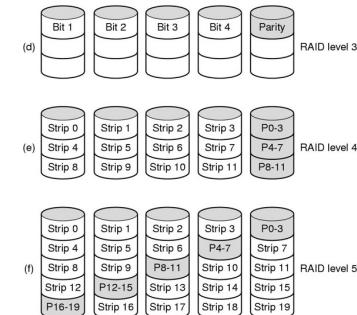


Sistemas Operativos - 2022/2023

222

.222

Sistemas RAID



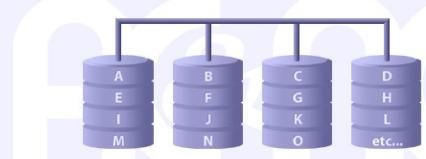
Sistemas Operativos - 2022/2023

223

.223

Sistemas RAID

RAID LEVEL 0 : Striped Disk Array without Fault Tolerance

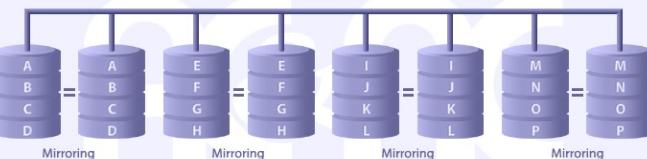


Copyright © 1996 - 2004 Advanced Computer & Network Corporation. All Rights Reserved.

.224

Sistemas RAID

RAID LEVEL 1 : Mirroring & Duplexing

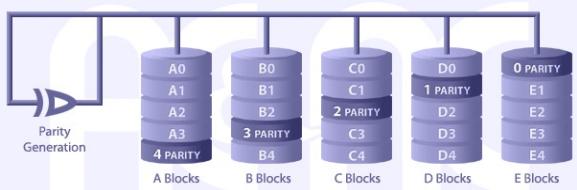


Copyright © 1996 - 2004 Advanced Computer & Network Corporation. All Rights Reserved.

.225

Sistemas RAID

RAID LEVEL 5 : Independent Data Disks with Distributed Parity Blocks



Copyright © 1996 - 2004 Advanced Computer & Network Corporation. All Rights Reserved.

.226

E se há um *crash* do sistema?



.227

Log-structured File Systems

- Devido à existência de caches em memória, e a necessidade de várias escritas em disco, há hipótese da informação ficar incoerente após crash => corrupção do SF
- FSCK pode demorar muito tempo pois tem de testar todos os meta-dados (faça man fsck e imagine os algoritmos)
 - **inaceitável** em certos cenários
- É preciso que o sistema de ficheiros **recupere depressa**

Solução?

Sistemas Operativos - 2022/2023

228

.228

Log-structured File Systems

- A solução passa por utilizar as “boas práticas” dos sistemas de gestão de Bases de Dados...
- SGBDs há muito utilizam **Logs** para garantir as propriedades ACID (aqui interessa em particular a Atomicidade)
 - SGBD escrevem no Log operações e dados
 - FS tendem a escrever apenas meta-dados (i-nodes, free block allocation maps, i-nodes maps, etc.)

Sistemas Operativos - 2022/2023

229

.229

Log-structured File Systems

- Os sistemas de ficheiros baseados em “diário” (Log) mantêm um registo (log) das operações de actualização do SF.
 - As transacções são registadas no Log
 - Em background, as operações indicadas no Log são executadas sobre o sistema de ficheiros e a transacção marcada como committed. Em caso de crash, reexecuta-se apenas o log não completado
 - Checkpointing pode atrasar aplicações
 - Numa leitura, se o bloco pretendido não tiver sido alvo de “checkpoint” há que consultar o log => atraso.

Sistemas Operativos - 2022/2023

230

.230

Estudo de casos

- MS-DOS
 - Baseado em FATs
 - File Allocation Tables indicam blocos ocupados por cada ficheiro e ainda os blocos livres na partição
 - Entrada na directória indica o primeiro bloco do ficheiro. Para localizar o seguinte é preciso seguir a FAT
 - Dimensão da FAT ? Pode obrigar a overlays de partes da FAT
 - Duplicação de FATs para tolerar corrupção

Sistemas Operativos - 2022/2023

231

.231

Estudo de casos

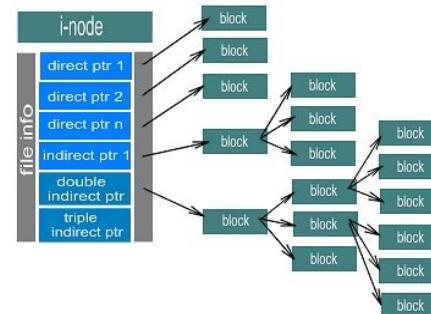
- Unix
 - Directorias + I-nodes + data blocks
 - Directorias
 - São ficheiros especiais que fazem a associação nome / i-node
 - I-nodes contêm restantes atributos dos ficheiros, incluindo permissões (ugo), datas e localização dos blocos (até 3 níveis de indirecção)

Sistemas Operativos - 2022/2023

232

.232

Estudo de casos

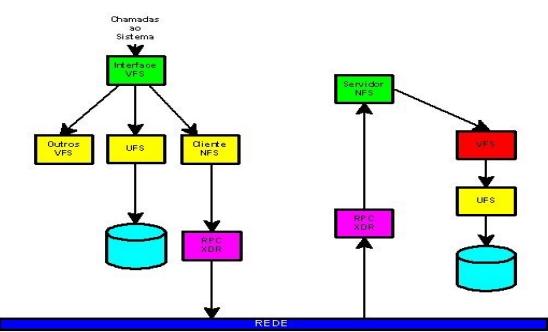


Sistemas Operativos - 2022/2023

233

.233

Network File System (NFS)



Sistemas Operativos - 2022/2023

234

.234

E ainda...

- Extent-based file systems
 - Parallel File Systems
 - Distributed File Systems
 - Storage Area Networks
- ...

Sistemas Operativos - 2022/2023

235

.235

O “estado-da-arte”

- Perquise no Google por Ext3, XFS, JFS, NTFS, Coda...
- Ou passe algum tempo em
<http://www.aspsys.com/software/links.aspx/14.aspx>
- Se o tempo é limitado, recomenda-se a leitura de
 - **Reiser FS** (<http://www.namesys.com/>)

Backups

- Assegure-se que percebe a diferença entre
 - Backup
 - Redundância nos discos, por exemplo Raid-1(mirroring) ou Raid-5
- Backups
 - Para onde? Quando? Que garantias de integridade?

FIM