

GSD
Grupo de Sistemas Distribuídos

Sistemas Operativos

MIEI

Grupo de Sistemas Distribuídos
fsm@di.uminho.pt

Sistemas Operativos - 2018/2019

5

GSD
Grupo de Sistemas Distribuídos

Equipa Docente

LSDS @ HASLab

- Responsável + Ts
fsm@di.uminho.pt
- PLs
Ana, Fábio, fsm, João Paulo, João Marco, psa, rco?
- Horário de atendimento a definir (use e-mail da Uminho!)

Sistemas Operativos - 2018/2019

6

GSD
Grupo de Sistemas Distribuídos

As más notícias

- É uma UC de engenharia:
 - É preciso usar a “massa cinzenta”
 - Também é preciso a “sujar as mãos”

=> programar, testar, configurar, debug, mexer no programa e ver o que dá...

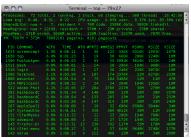



- Não basta ir ao Google ou “ver” os exercícios resolvidos por outros

Sistemas Operativos - 2018/2019

7

GSD
Grupo de Sistemas Distribuídos

- Má preparação anterior tem de ser recuperada ASAP!
 - concepção de algoritmos e estruturas de dados, programação em C, pouco tempo a BATER CÓDIGO, pouco trabalho individual...
- Interface ”linha de comando”
 
- Guiões vão-se acumulando:
Ficheiros + Processos + Redir + pipes + + FIFOs + Sinais
 

Sistemas Operativos - 2018/2019

8

As boas notícias

- Apesar da “mística” que rodeia esta unidade curricular...
 - basta um pouco de trabalho* e bom senso para fazer esta unidade curricular
 - e ultrapassar grande parte das limitações anteriores!

(*) Participar nas aulas **+ 2 a 3 horas POR SEMANA** (sem contar com a recuperação de dificuldades anteriores)

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 fazer?
 - Foi abaixo... Perdi tudo?

Numa palavra...

- Se fosse necessário dizer de que trata esta UC de Sistemas Operativos:
 - Numa só palavra:
CONCORRÊNCIA
 - Em mais do que uma:
 - Concorrência, concorrência...
 - Eficiência, rapidez, segurança, etc.
 - Boa gestão de recursos perante determinada carga
 - Perceber como funcionam os sistemas (Apps, SO, HW)

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
 - Nada de janelas, nem ratos...
 - Terminal com *Bash*, comandos, pipelines...
 - Programação de “baixo nível”: C, syscalls, libs...





Bibliografia recomendada

- Silberschatz, Galvin and Gagne, *Operating System Concepts*, John Wiley & Sons, 2010.
- Carlos Ribeiro, Alves Marques, 2^a-ed, FCA Editora Informática, 2012.
- FSM 2004, VOU FAZER SISTEMAS OPERATIVOS**

Sistemas Operativos - 2018/2019

13



Bibliografia Adicional

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

Sistemas Operativos - 2018/2019

14



Slides

- Baseados os originais que acompanham os livros recomendados (em especial Silberschatz)
- (Progressivamente) disponíveis no Blackboard
- Servem apenas de “âncora” ao estudo
→ Não chegam para responder aos testes!!!

Sistemas Operativos - 2018/2019

15



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 pensar:
O que é que EU aprendi com este exercício?
- Recomenda-se vivamente:
 - estudar o guião antes da respectiva aula
 - usar a aula para tirar dúvidas e não para copiar o que está no quadro, no colega do lado, no Google...
 - terminar em casa todas as alíneas não resolvidas durante a aula; se necessário, peça ajuda por mail

Sistemas Operativos - 2018/2019

16



Avaliação

- 1/3 **Trabalho Prático** em grupo (<=3 elementos)
- 2/3 **Prova escrita** (teste individual e/ou exame de recurso)
- Há nota mínima no TP (10) e no teste/exame (9)
- **É obrigatório ter nota positiva no TP.** Sem essa nota, terá de voltar no ano seguinte.
- Nota do TP do ano anterior pode ser “reutilizada” mas é truncada a 15 valores
(Atenção: só pode usar do **ano lectivo anterior**)

Sistemas Operativos - 2018/2019

17



Avaliação

```
nota_final (tp, t)
{
    if (tp < 10) return (NA);
    if (t >= 9) return (1/3*tp + 2/3*t)
    return (A) // tem de ir ao exame de recurso
    // no exame, o A passa a R.
}
```

Sistemas Operativos - 2018/2019

18



Avaliação

Prova escrita (teste ou exame) individual e sem consulta.

- É preciso escrever código que resolva **o problema proposto**; não basta reproduzir os guões
- 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.

Ninguém faz esta UC apenas com a parte teórica
Dificilmente a fará só com a prática

Sistemas Operativos - 2018/2019

19



Programa

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

Sistemas Operativos - 2018/2019

20



Is it harder...

- Is it harder to become a systems programming engineer than other types of software engineer?
- Another way to look at this questions is: **is it harder to work on a operating system than on a website?**

Sistemas Operativos - 2018/2019

YES...

- Systems Programming is "working without a net" in that when an application programmer screws up, his application crashes, but the computer system keeps on operating. When a systems programmer screws up, the computer system crashes en toto (halts).
- Systems programmers must know how a computer actually works, i.e. know something about Computer Architecture, Compilers...
- Systems programming deals with Concurrency all the time
- Systems Programming is lower level.

Sistemas Operativos - 2018/2019



Vamos começar pelo princípio...

- Para que serve um computador?
 - Para executar programas (aplicações)
 - Que facilitam a vida aos utilizadores

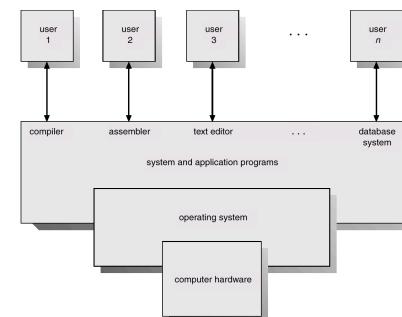
Sistemas Operativos - 2018/2019

23



O que é um Sistema Operativo?

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



Sistemas Operativos - 2018/2019

24

Portanto...

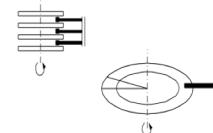
- SO deve colocar o hardware à disposição dos programas e utilizadores, mas de uma forma
 - **conveniente,**
 - **protegida,**
 - **eficiente,**
 - **justa,**
 - ...

Sistemas Operativos - 2018/2019

25

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 - 2018/2019

26

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 - 2018/2019

27

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 - 2018/2019

28

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 - 2018/2019

29

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 - 2018/2019

30

Evolução



- Sistemas de Computação
 - 1^a geração (1945/1955) – Válvulas
 - 2^a geração (1955/1965) – Transistores, *batch*
 - 3^a geração (1965/1980) – ICs, *time-sharing*
 - 4^a geração (1980/ ?) – PCs, workstations, ethernet, modelo Cliente/Servidor
“The network IS the system”
- Hoje temos...
 - Datacenters, servidores, portáteis, tablets, smartphones...
 - Tudo interligado em “Cloud”
 - Internet of things (IOT)...

Sistemas Operativos - 2018/2019

31

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 - 2018/2019

GSD
Grupo de Sistemas Distribuídos

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 - 2018/2019

33

GSD
Grupo de Sistemas Distribuídos

No início era assim...

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



- 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...

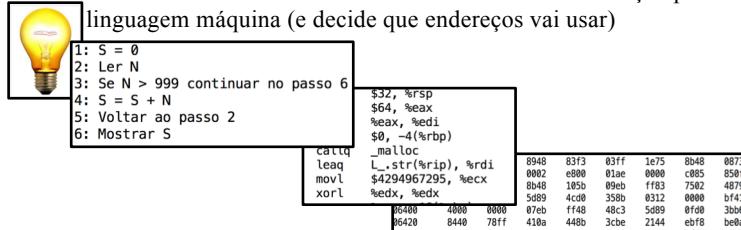
→ Sim, TUDO!!!!

Sistemas Operativos - 2018/2019

GSD
Grupo de Sistemas Distribuídos

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 e faz então a tradução para linguagem máquina (e decide que endereços vai usar)



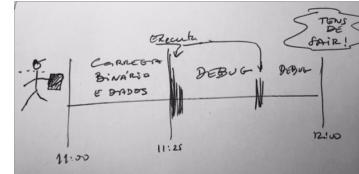
Endereço	Valor
8948	83f3 83ff 1e75 8b48 0073
0082	e000 01ae 0000 c885 85bf
8848	105b 09eb ff83 7502 4879
5d89	4cd0 358b 0312 0000 b741
06400	4000 0000 07eb ff48 48c3 5d89 0fd0 3bb6
06420	8440 78ff 410a 44b8 3cbe 2144 efd8 be0a
06440	4000 0000 55e8 0001 8500 75c0 0fd9 03b6

Sistemas Operativos - 2018/2019

GSD
Grupo de Sistemas Distribuídos

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



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

Sistemas Operativos - 2018/2019

GSD
Grupo de Sistemas Distribuídos

- [HP 2114B \(1968\)](#)
- [Altair 8800 \(1975\)](#)

Sistemas Operativos - 2018/2019

GSD
Grupo de Sistemas Distribuídos

E para aumentar a eficiência...

- Introduziu-se um operador especializado
 - Utilizador entrega fita perfurada ou cartões
 - Operador carrega o programa, executa-o e devolve os resultados
- **Ganhou-se em eficiência, perdeu-se em conveniência**
 - Operador é especialista em operação, não em programação
 - Pode haver escalonamento (i.e. alteração da ordem de execução)
 - Utilizador deixou de interagir com o seu programa

Sistemas Operativos - 2018/2019

38

GSD
Grupo de Sistemas Distribuídos

Melhor do que ter um operador...

- É automatizar, ter um programa que:
 - Controla a operação do computador
 - Encadeia “jobs”, operador apenas carrega e descarrega
- Utilizadores devem usar rotinas de IO do sistema (embora ainda possam escrever as suas)

Embrião de um sistema operativo?

Sistemas Operativos - 2018/2019

39

GSD
Grupo de Sistemas Distribuídos

Mas havia o risco de...

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

Sistemas Operativos - 2018/2019

40

GSD
Grupo de Sistemas Distribuídos

Soluções (hardware)

- Interrupções
- Relógio de Tempo Virtual
- Instruções privilegiadas, 2 ou mais modos de execução
- Protecção de memória

Sistemas Operativos - 2018/2019

41

GSD
Grupo de Sistemas Distribuídos

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*

Sistemas Operativos - 2018/2019

42

GSD
Grupo de Sistemas Distribuídos

Exemplo: Interrupt-driven IO

(a)

(b)

(a) OS inicia operação de IO e prepara-se para receber a interrupção; entretanto pode ir executando 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.

Sistemas Operativos - 2018/2019

43

GSD
Grupo de Sistemas Distribuídos

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

resident monitor

① trap to monitor

② perform I/O

③ return to user

user program

case n

⋮

read

⋮

system call n

⋮

Sistemas Operativos - 2018/2019

44

GSD
Grupo de Sistemas Distribuídos

Primeiros sistemas de batch

(a) (b) (c) (d) (e) (f)

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

Sistemas Operativos - 2018/2019

45

GSD
Grupo de Sistemas Distribuídos

Exemplo de um “job”

\$END
Data for program
\$RUN
\$LOAD
Fortran Program
\$FORTRAN
\$JOB, 10,6610802, MARVIN TANENBAUM

Sistemas Operativos - 2018/2019

46

GSD
Grupo de Sistemas Distribuídos

Multiprogramação

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

0
monitor
256000
job 1
380040
job 2
420940
job 3
880000
job 4
1024000

base register: 300040
limit register: 120900

Sistemas Operativos - 2018/2019

47

GSD
Grupo de Sistemas Distribuídos

Protecção de memória

CPU → address → base → diamond 'a' → if yes: trap to operating system monitor-addressing error; if no: diamond 'c' → if yes: memory; if no: trap to operating system monitor-addressing error

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 - 2018/2019

48

GSD
Grupo de Sistemas Distribuídos

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 - 2018/2019

49

GSD
Grupo de Sistemas Distribuídos

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

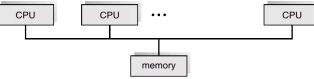
Sistemas Operativos - 2018/2019

50

GSD
Grupo de Sistemas Distribuídos

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, dividida em vários processos

Sistemas Operativos - 2018/2019

51

GSD
Grupo de Sistemas Distribuídos

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 Operativos - 2018/2019

52

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 Operativos - 2018/2019

53

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...



54

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 - 2018/2019

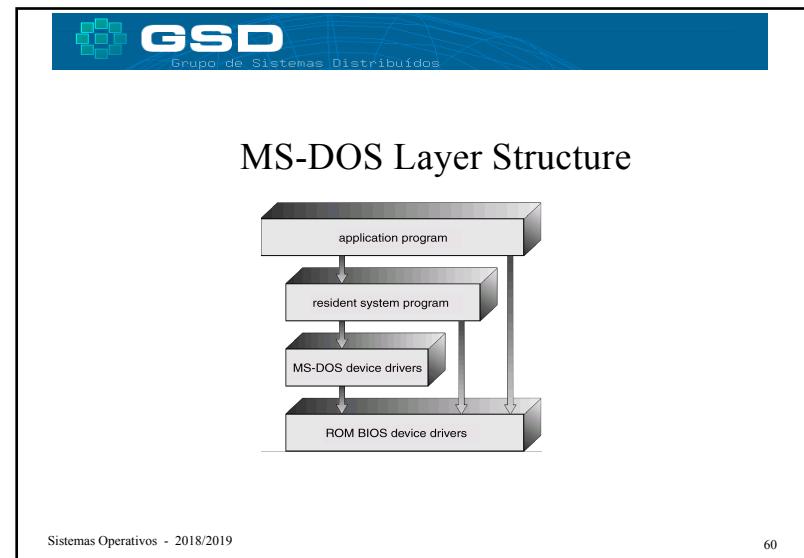
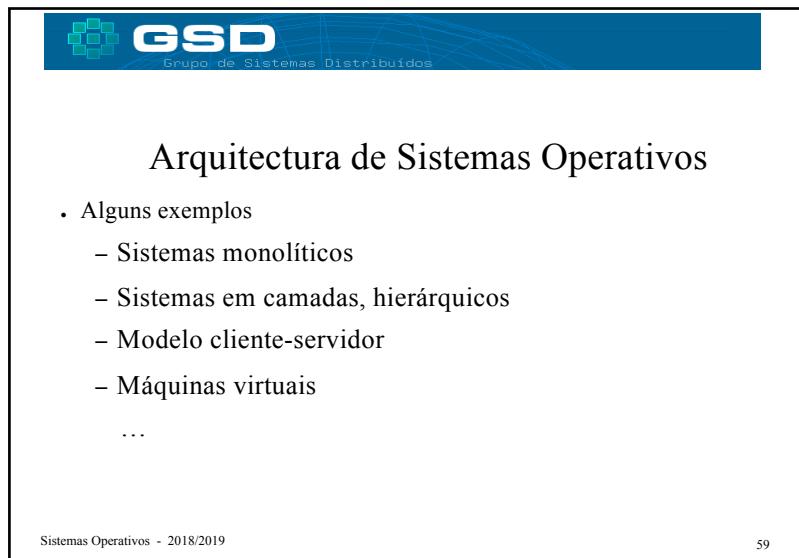
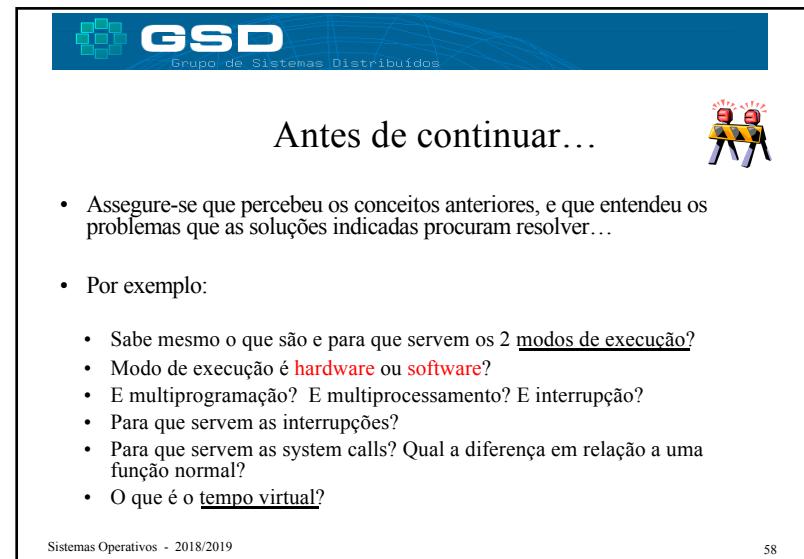
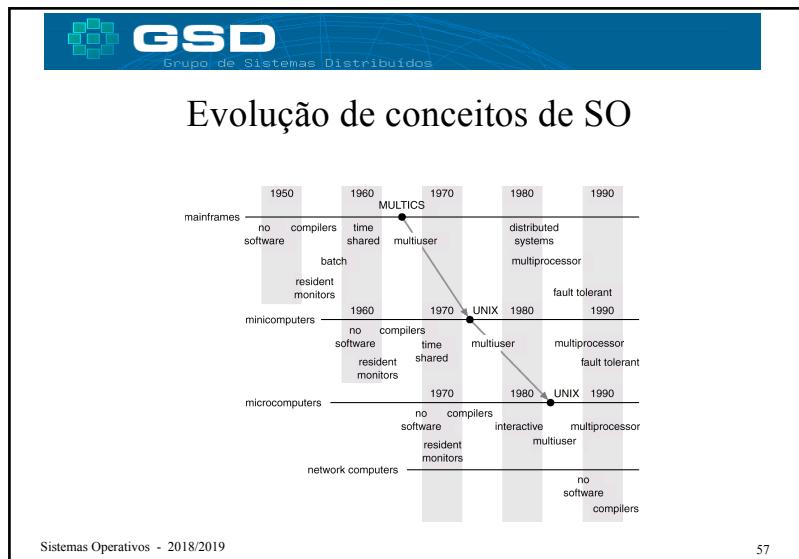
55

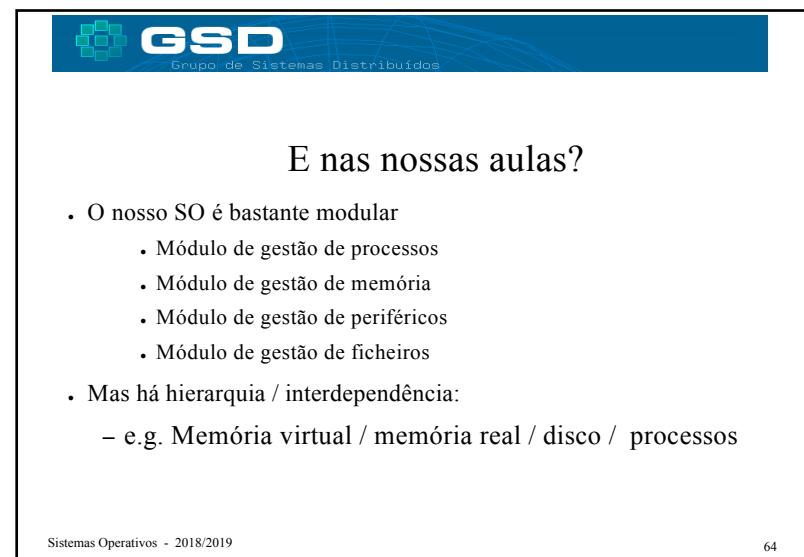
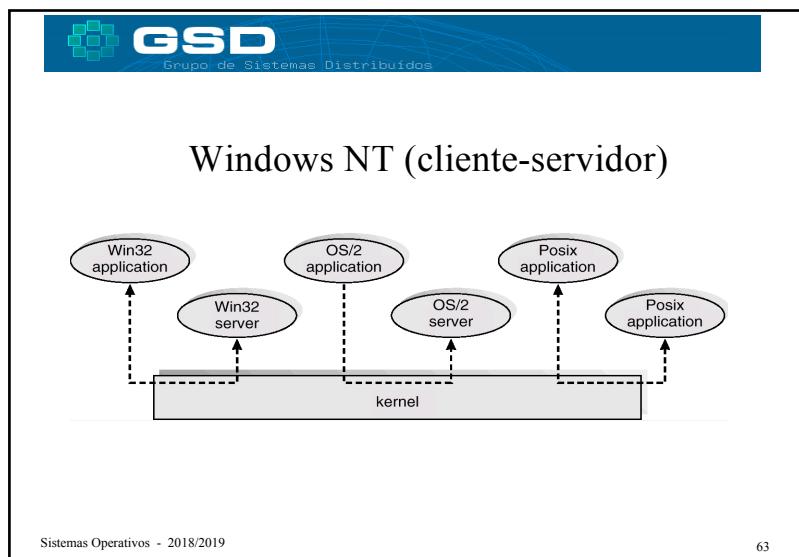
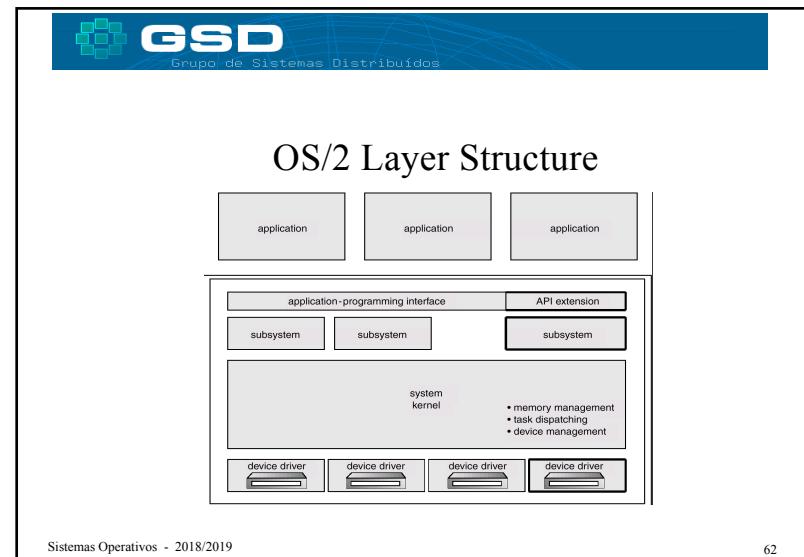
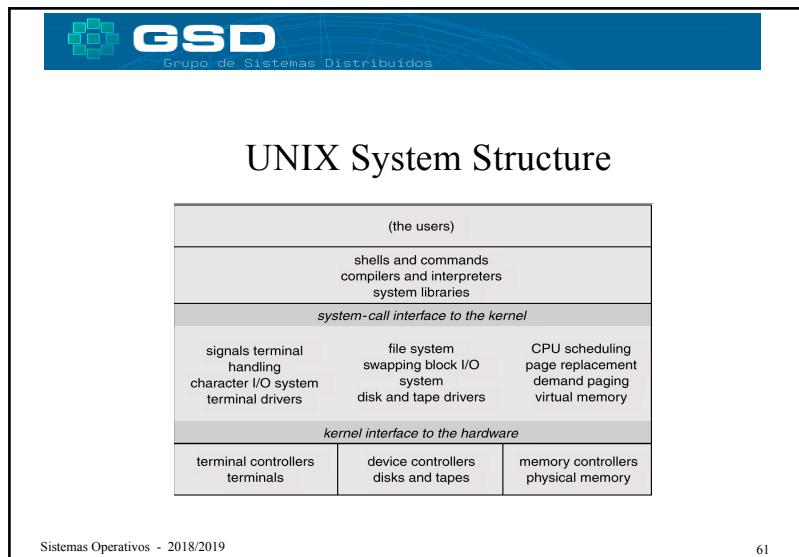
E ainda...

- 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...



56





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...

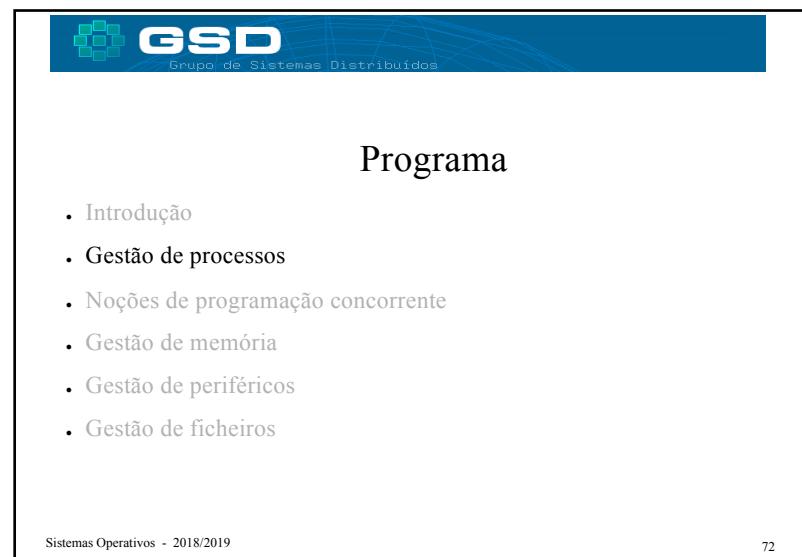
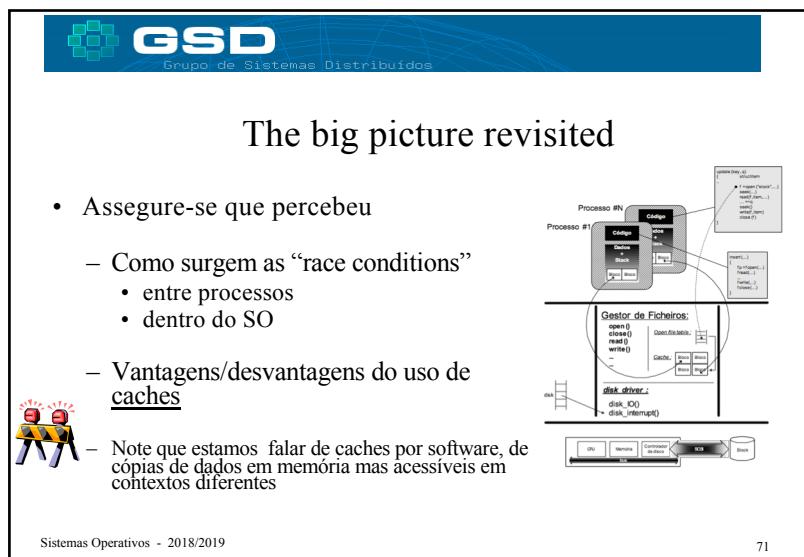
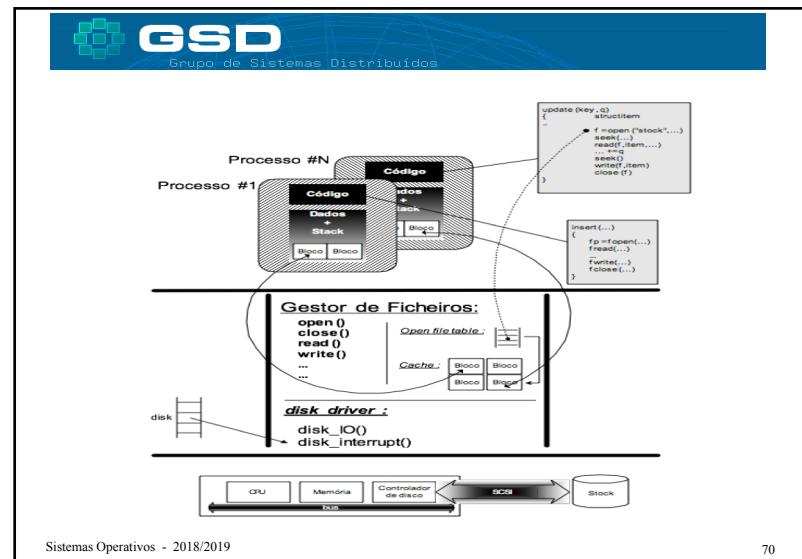
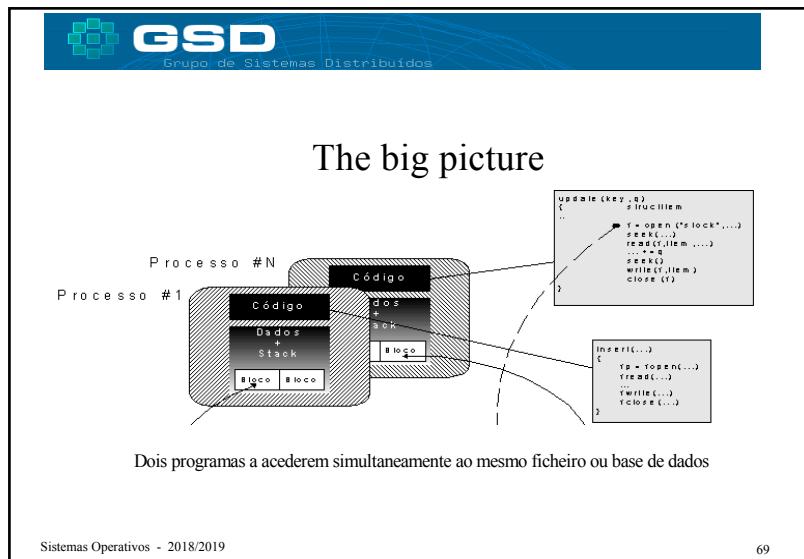
- Sabemos de facto
 - “Como é” um programa (e porquê?)
 - “Como é” um computador (e porquê?)
- Ou seja,
 - perceber as razões para o hardware e software de sistemas serem como são

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...)

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!)



Porquê criar vários processos?

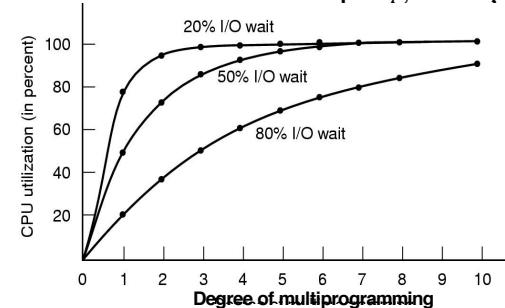
- Porque dá jeito... + conveniência
 - Estruturação dos programas
 - Para não estar à espera (spooling, background...)
 - Múltiplas actividades / janelas

- Porque é melhor + eficiência
 - Múltiplos CPUs
 - Aumenta a utilização de recursos (e.g multiprogramação)

Sistemas Operativos - 2018/2019

73

Benefícios da multiprogramação



Sistemas Operativos - 2018/2019

74

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.)

Sistemas Operativos - 2018/2019

75

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, ...

Sistemas Operativos - 2018/2019

76

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		

Sistemas Operativos - 2018/2019

77

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”

Sistemas Operativos - 2018/2019

78

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

Sistemas Operativos - 2018/2019

79

Políticas de escalonamento

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

Sistemas Operativos - 2018/2019

80

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.

Sistemas Operativos - 2018/2019

81

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...

Sistemas Operativos - 2018/2019

82

Estados de um processo (i)



Sistemas Operativos - 2018/2019

83

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

Sistemas Operativos - 2018/2019

84

GSD
Grupo de Sistemas Distribuídos

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
    */
}
```

Sistemas Operativos - 2018/2019

85

GSD
Grupo de Sistemas Distribuídos

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.

Sistemas Operativos - 2018/2019

86

GSD
Grupo de Sistemas Distribuídos

Estados de um processo (ii)

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

```
graph LR; A((Em execução)) --> B((Bloqueado)); B --> A;
```

Sistemas Operativos - 2018/2019

87

GSD
Grupo de Sistemas Distribuídos

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

```
graph TD; A((Pronto)) --> B((Em execução)); A --> C((Bloqueado)); B --> C;
```

Sistemas Operativos - 2018/2019

88

Estados de um processo (iv)

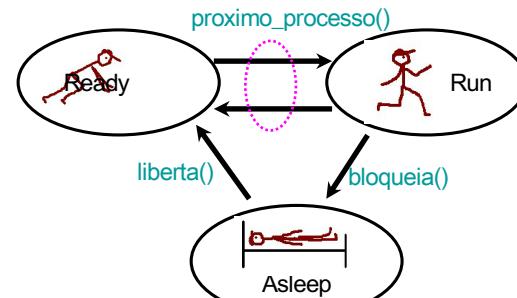
- Em execução
 - Foi-lhe atribuído o/um CPU, executa o programa correspondente
- Bloqueado
 - O processo está logicamente impedido de prosseguir, e.g. porque lhe falta um recurso ou espera por evento
 - Do ponto de vista do SO, é uma transição **VOLUNTÁRIA!**
- Pronto a executar, aguarda escalonamento



Sistemas Operativos - 2018/2019

89

Primitivas de despacho (i)



Sistemas Operativos - 2018/2019

90

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()**



Sistemas Operativos - 2018/2019

91

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 - 2018/2019

92

Principais decisões

- Qual o próximo processo?
- Quando começa a executar?
- Durante quanto tempo?

- Por outras palavras,

Há **desafectação forçada** ou não?

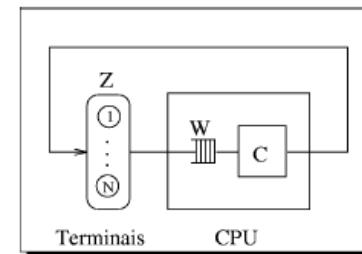
Escalonamento de processos

- 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)

Escalonamento de processos

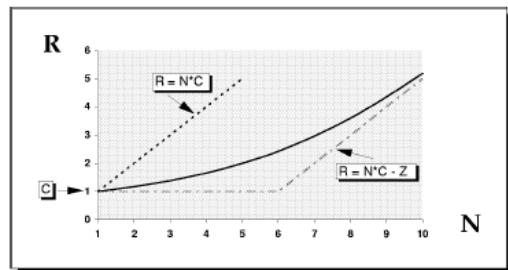
- Escalonamento **cooperativo** (non-preemptive).
 - “poor man’s approach to multitasking” ?
 - Sensível às variações de carga
- Escalonamento com **desafectação forçada**
 - Sistema “responde” melhor
 - Mas a comutação de contexto tem overhead

Modelo de sistema interactivivo



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

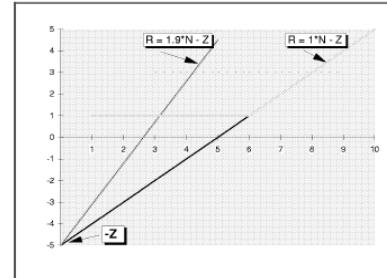
Tempo de Resposta (carga homogénea)



Sistemas Operativos - 2018/2019

97

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

Sistemas Operativos - 2018/2019

98

Tempo de Resposta (carga heterogénea)

- Para evitar que as interacções 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.

Sistemas Operativos - 2018/2019

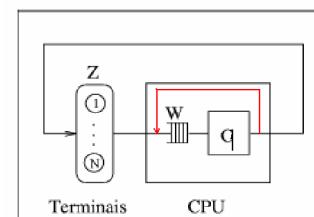
99

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!

$$R = W + q + W + c'$$



Sistemas Operativos - 2018/2019

100

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)

Escalonamento de processos

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

Escalonamento de processos

- A decisão de escalarar 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

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

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)

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



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 - 2018/2019

109



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 - 2018/2019

110



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.

Sistemas Operativos - 2018/2019

111



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

Sistemas Operativos - 2018/2019

112

Preemptive Priority

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

RR (Round Robin)

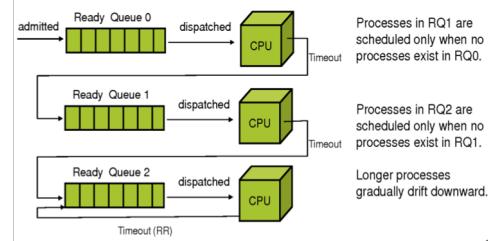
- Dá a cada processo um intervalo de tempo fixo de CPU de cada vez
- Quando um processo esgota o seu quanto retira-o do CPU e volta a colocá-lo no fim da fila.
- Ignorando os overheads do escalonamento, cada um dos n processos CPU-bound terá $(1/n)$ do tempo disponível de CPU

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)

Multilevel Feedback Queue Scheduling

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



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

Sistemas Operativos - 2018/2019

117

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

Sistemas Operativos - 2018/2019

118

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 - 2018/2019

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 - 2018/2019



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 - 2018/2019



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 - 2018/2019



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 - 2018/2019



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 - 2018/2019



Exemplos de concorrência

- Inspirados na vida real
 - Diálogo cliente/bar(wo)men
 - Acesso ao WC
 - Acesso a um 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 - 2018/2019



Sincronização

BARMAN

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

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

CLIENTE

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

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

Sistemas Operativos - 2018/2019



Estará correcto?

BARMAN

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

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

CLIENTE

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

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

Sistemas Operativos - 2018/2019



Claro que não está...

- Existem esperas activas => desperdício de CPU
- Não garante que não haja copos partidos
- Nem clientes a pegarem no mesmo copo...

Sistemas Operativos - 2018/2019



Mecanismos de sincronização

- Existem muitos
 - Semáforos, mensagens, pipelines...
- Nas aulas teóricas vamos usar semáforos, mostrando a dualidade com “mensagens” (no nosso caso são pipelines do Unix)

Sistemas Operativos - 2018/2019



Semáforos

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

* Na realidade há mais operações (e.g. Trylock)

Sistemas Operativos - 2018/2019



Semáforos

- Imagine uma caixa com bolas, rebuçados, pedras...
- E as operações seguintes:
 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

Nota: pense que P pode PARAR e V pode acordar um processo parado (VAI...)

Sistemas Operativos - 2018/2019



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 - 2018/2019

GSD
Grupo de Sistemas Distribuídos

Semáforos

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

Sistemas Operativos - 2018/2019

GSD
Grupo de Sistemas Distribuídos

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 - 2018/2019

GSD
Grupo de Sistemas Distribuídos

<p>BARMAN</p> <pre>/* aguarda por vaga no balcão */ /* e só depois vai... */ pousar_copo_no_balcao(); /* avisa que há +1 copo cheio */ V(copo)</pre>	<p>CLIENTE</p> <pre>/* aguarda por copo cheio */ P(copo) tirar_copo_do_balcao(); /* avisa que há vaga no balcão */</pre>
--	---

Falta inicializar o semáforo copo a Zero

Sistemas Operativos - 2018/2019

GSD
Grupo de Sistemas Distribuídos

Capacidade

- 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
 - Equivale a inicializar o semáforo a 0 e de imediato executar tantos V() quantas as posições livres no balcão

Sistemas Operativos - 2018/2019

GSD
Grupo de Sistemas Distribuídos

BARMAN	CLIENTE
<pre>/* aguarda por vaga no balcão */ P(espaço) pousar_copo_no_balcao(); /* avisa que há +1 copo cheio */ V(copo) ...</pre>	<pre>/* aguarda por copo cheio */ P(copo) tirar_copo_do_balcao(); /* avisa que há vaga */ V(espaço)</pre>
Falta inicializar o semáforo copo a <u>Zero</u> e espaço à capacidade do balcão	

Sistemas Operativos - 2018/2019

GSD
Grupo de Sistemas Distribuídos

Exclusão mútua

- Exclusão mútua é também uma forma de “sincronização”
- Talvez aqui a palavra seja **(des)sincronização**, pois queremos garantir que 2 ou mais processos não estão simultaneamente dentro da região crítica...
- Esqueleto da solução
 - Entrada: /* espera por região livre, e ocupa-a */
 - Código correspondente à região crítica
 - Saída: /* liberta região*/

Sistemas Operativos - 2018/2019

GSD
Grupo de Sistemas Distribuídos

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 - 2018/2019

GSD
Grupo de Sistemas Distribuídos

“Receitas” com semáforos

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

Sistemas Operativos - 2018/2019



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 barman + N clientes)
- =>Falta garantir **exclusão mútua** no acesso ao balcão!

Sistemas Operativos - 2018/2019



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.

Sistemas Operativos - 2018/2019



Programa

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

Sistemas Operativos - 2018/2019

143

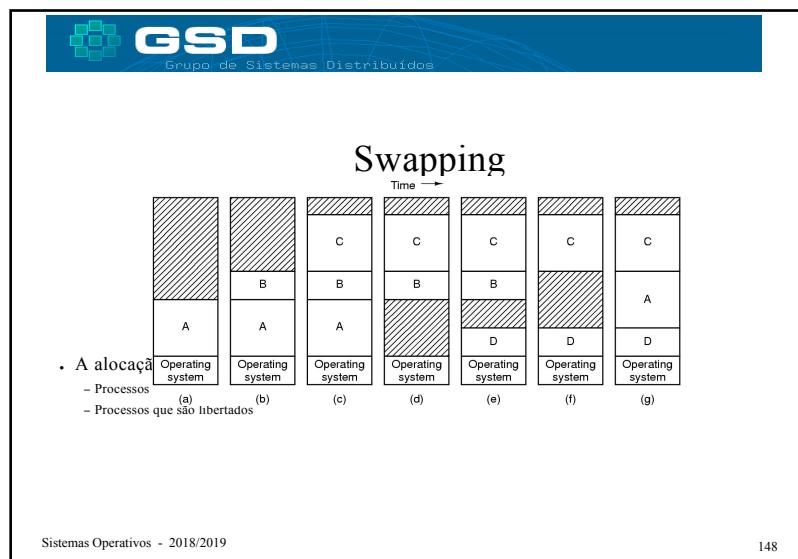
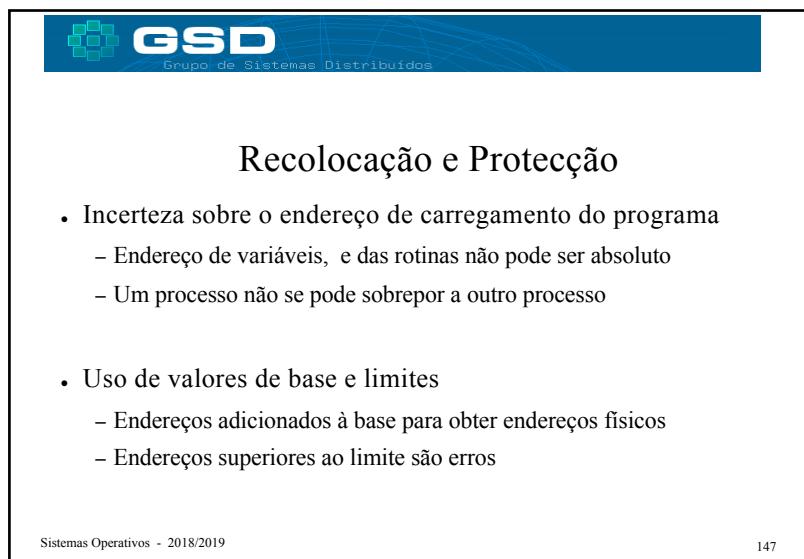
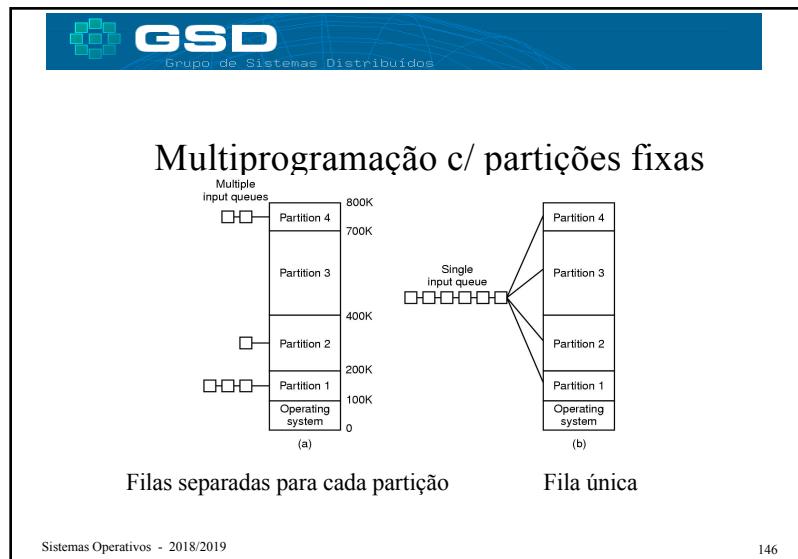
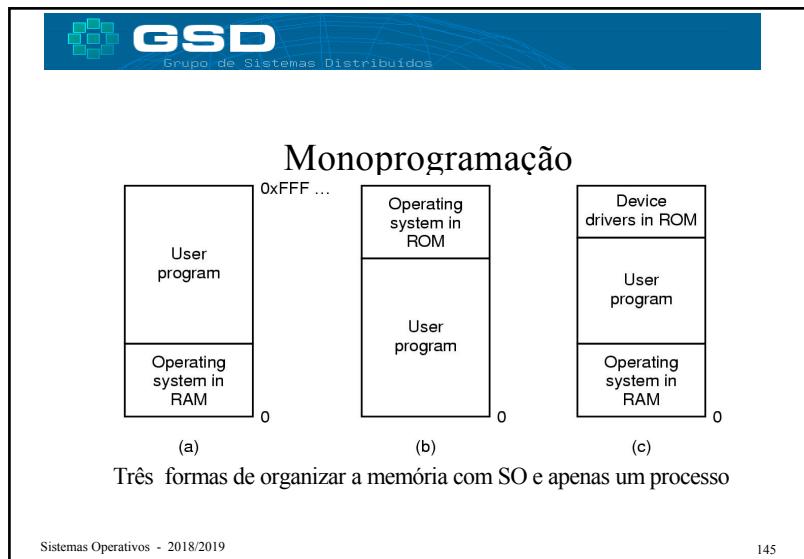


Gestão de Memória

- Idealmente a memória seria:
 - grande
 - rápida
 - não volátil
- Hierarquia da memória
 - Pouca memória rápida, cara – cache
 - Velocidade média, custo aceitável – memória principal
 - Gigabytes de memória lenta, discos baratos
- O gestor de memória gera esta hierarquia da memória

Sistemas Operativos - 2018/2019

144



GSD
Grupo de Sistemas Distribuídos

Swapping

The diagram shows two configurations of memory:

- (a) Process A is swapped out. It contains an operating system, program, and data. A stack is shown above the program. The memory is divided into regions: 'Room for growth' at the top and bottom, and 'Actually in use' in the middle.
- (b) Process B is swapped in. It also contains an operating system, program, and data, along with its own stack. Similar 'Room for growth' and 'Actually in use' regions are shown.

- Alocação para swap
- Alocação para swap

149

GSD
Grupo de Sistemas Distribuídos

Gestão de memória com bitmaps

The diagram illustrates the management of memory zones (A-E) and processes (P0-P14) using bitmaps:

- (a) Shows memory zones A through E with their respective starting addresses and sizes: A (8), B (16), C (16), D (24), and E (8).
- (b) Shows the bitmap corresponding to the memory zones. A hole starts at address 18 with a length of 2.
- (c) Shows a linked list of processes (P0-P14) with their start addresses and lengths, connected by arrows.

- Zona de memória
- Bitmap correspondente
- Mesma informação de uma lista ligada

150

GSD
Grupo de Sistemas Distribuídos

Gestão de memória listas ligadas

	Before X terminates	After X terminates
(a)	A X B	becomes A B
(b)	A X	becomes A
(c)	X B	becomes B
(d)	X	becomes

Quatro cenários para a terminação do processo X

Sistemas Operativos - 2018/2019

151

GSD
Grupo de Sistemas Distribuídos

Memória Virtual

The diagram shows the flow of virtual memory management:

- The CPU sends virtual addresses to the MMU (Memory Management Unit).
- The MMU sends physical addresses to the memory.
- The memory, disk controller, and bus are interconnected.

Sistemas Operativos - 2018/2019

152

GSD
Grupo de Sistemas Distribuídos

Memória Virtual

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

Sistemas Operativos - 2018/2019

153

GSD
Grupo de Sistemas Distribuídos

Memória Virtual

Operação da MMU com 16 páginas de 4 KB

Sistemas Operativos - 2018/2019

154

GSD
Grupo de Sistemas Distribuídos

Memória Virtual

- Endereço de 32 bit c/ 2 campos para tabelas de páginas
- Tabelas de páginas de 2 níveis

Sistemas Operativos - 2018/2019

155

GSD
Grupo de Sistemas Distribuídos

Memória Virtual

Entrada típica da tabela de páginas

Sistemas Operativos - 2018/2019

156

GSD
Grupo de Sistemas Distribuídos

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

Sistemas Operativos - 2018/2019 157

GSD
Grupo de Sistemas Distribuídos

Memória Virtual

The diagram illustrates the difference between traditional page tables and inverted page tables. On the left, a traditional page table is shown as a vertical array of 2⁵² entries, indexed by virtual page number. In the middle, physical memory is represented as a vertical array of 2¹⁶ 4-KB page frames. On the right, an inverted page table is shown as a hash table where each entry points to a page frame, indexed by virtual page number.

- Comparação entre tabelas tradicionais e tabelas invertidas

Sistemas Operativos - 2018/2019 158

GSD
Grupo de Sistemas Distribuídos

Aspectos de Implementação

Tratamento da Page Fault

1. MMU interrompe o processador.
2. Kernel salvaguarda os registos e invoca RTI
3. SO determina a página virtual necessária
4. SO valida endereço e procura/cria page frame
 - Page in, zero fill, in transit...

Sistemas Operativos - 2018/2019 159

GSD
Grupo de Sistemas Distribuídos

Aspectos de Implementação

O SO intervém 4 vezes na paginação:

1. Criação do processo
 - Determinar o tamanho do programa
 - Criar a tabela de páginas
2. Execução do processo
 - Re-inicializar a MMU para o novo processo
 - Limpar a TLB
3. Na Page Fault
 - Determinar o endereço virtual causador da page fault
 - Colocar a página em memória, se endereço for legal
4. Fim da execução do processo
 - Libertar a tabela de páginas e as páginas associadas

Sistemas Operativos - 2018/2019 160

GSD
Grupo de Sistemas Distribuídos

Segmentação

The diagram illustrates the virtual address space as a stack of segments. From bottom to top, the segments are: Symbol table, Source text, Constant table, Parse tree, and Call stack. A bracket on the left indicates the 'Address space allocated to the parse tree'. Another bracket on the right indicates 'Space currently being used by the parse tree'. A note at the bottom states: 'Symbol table has bumped into the source text table'.

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

Sistemas Operativos - 2018/2019 161

GSD
Grupo de Sistemas Distribuídos

Segmentação

The diagram shows five segments labeled Segment 0 through Segment 4. Each segment has a vertical scale from 0K to 20K or 16K. Segment 0 contains the 'Symbol table'. Segment 1 contains 'Source text'. Segment 2 contains 'Constants'. Segment 3 contains the 'Parse tree'. Segment 4 contains the 'Call stack'. A note at the bottom states: 'Cada tabela pode crescer ou encolher independentemente'.

Sistemas Operativos - 2018/2019 162

GSD
Grupo de Sistemas Distribuídos

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 - 2018/2019 163

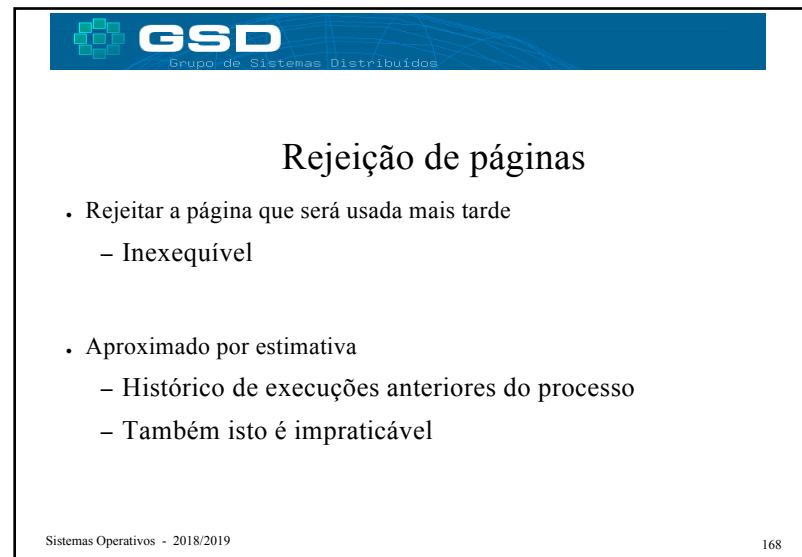
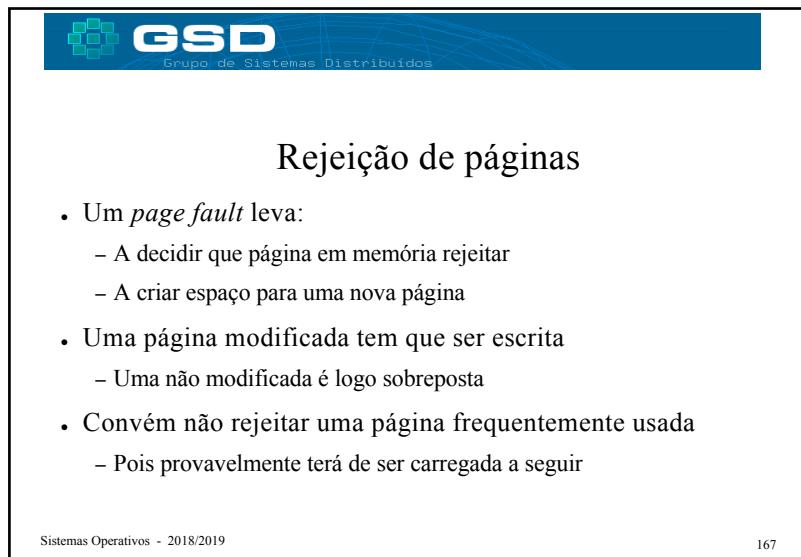
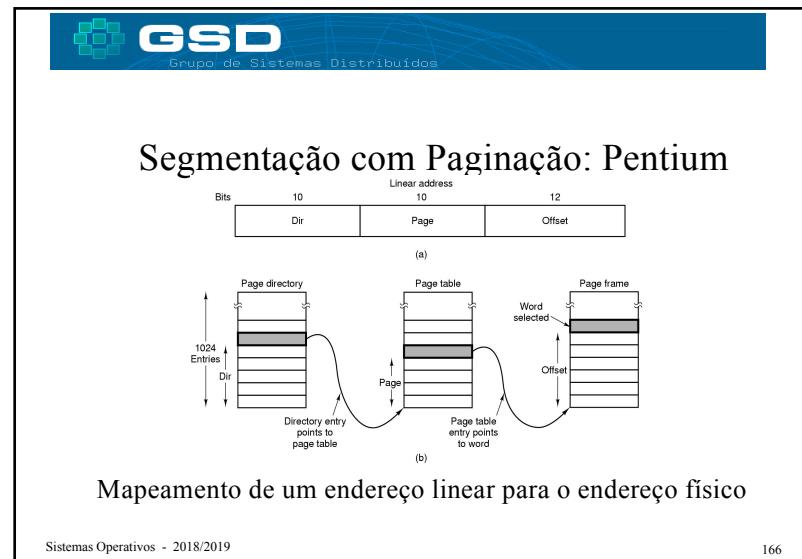
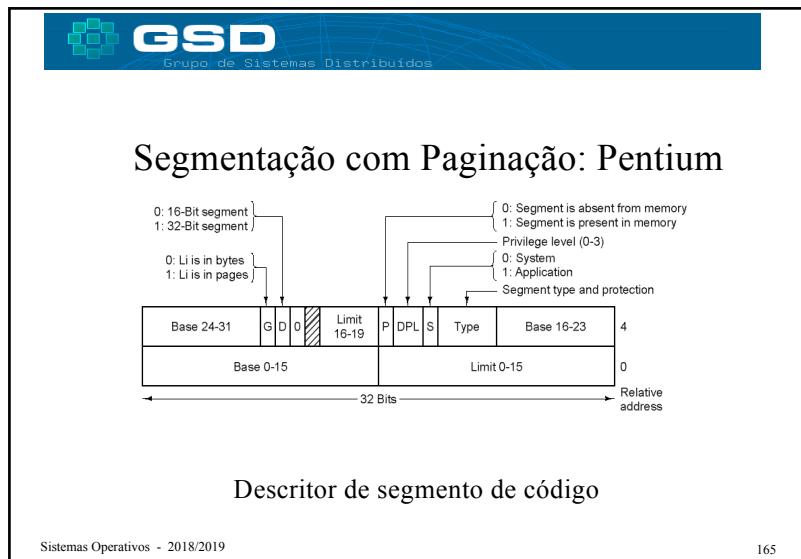
GSD
Grupo de Sistemas Distribuídos

Segmentação com Paginação: Pentium

The diagram shows a 13-bit selector register divided into three fields: Index (13 bits), 1 (1 bit), and 2 (2 bits). The 1 bit is labeled '0 = GDT/1 = LDT' and the 2 bits are labeled 'Privilege level (0-3)'.

Um **selector** no Pentium

Sistemas Operativos - 2018/2019 164



Rejeição de páginas NRU

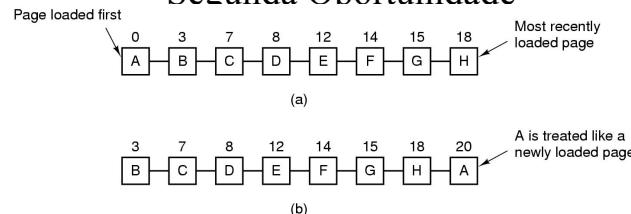
- Cada página tem 1 bit de acesso e 1 de escrita
- As páginas são assim classificadas:
 1. Não acedida, não modificada
 2. Não acedida, modificada
 3. Acedida, não modificada
 4. Acedida, modificada

NRU remove a página com menor “ranking”

Rejeição de páginas FIFO

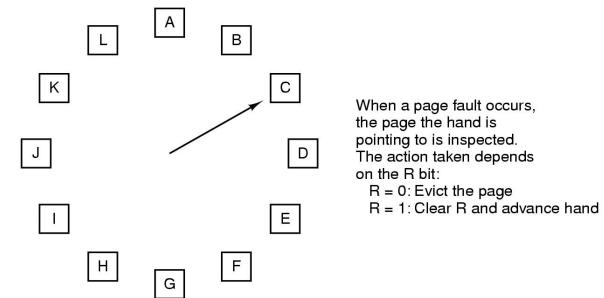
- Mantém uma lista das páginas em memória
 - Segundo a ordem em que foram carregadas
- A página no topo da lista é rejeitada
- Desvantagem
 - A página há mais tempo em memória poderá ser a mais usada

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

Rejeição de páginas: Relógio

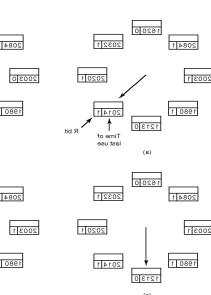


Rejeição de páginas LRU

- Assume que as páginas usadas recentemente serão usadas em breve
 - Rejeitar a página não usada há mais tempo
- Tem que gerir uma lista de páginas
 - Ordenada pela mais recente
 - Actualizada em todos os acessos à memória!
- Alternativamente manter um contador em cada entrada da tabela de páginas
 - Escolher a página com o menor valor
 - Periodicamente zerar o contador

WorkingSetClock

em laço interno



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.
WClock	Aproximação boa e eficiente.

Anomalia de Belady

All pages frames initially empty	
Youngest page	0 1 2 3 0 1 4 0 1 2 3 4
	0 1 2 3 0 1 1 1 1 2 3
Oldest page	0 1 2 3 0 0 0 1 4 4
	P P P P P P P P

(a) 9 Page faults

All pages frames initially empty	
Youngest page	0 1 2 3 0 3 4 0 1 2 3 4
	0 1 2 3 0 1 1 2 3 4 0 1 2 3
Oldest page	0 1 2 3 0 0 0 1 2 3 4 0 1 2 3
	P P P P P P P P P P P P P P

(b) 10 Page faults

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

GSD
Grupo de Sistemas Distribuídos

Sistemas Paginados

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

Sistemas Operativos - 2018/2019 177

GSD
Grupo de Sistemas Distribuídos

Alocacão 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) Config. original

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

(b) Sust. local

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

(c) Sust. global

Sistemas Operativos - 2018/2019 178

GSD
Grupo de Sistemas Distribuídos

Alocacão Local e Global

- Alocacão de páginas Local e Global

Page faults/sec

Número d

Number of page frames assigned

A

B

Sistemas Operativos - 2018/2019 179

GSD
Grupo de Sistemas Distribuídos

Controlo de carga / thrashing

- Apesar de um bom desenho, pode ainda ocorrer **thrashing**
- Quando a Frequência de Page Faults indica que:
 - Alguns processos precisam de mais memória central
 - Mas nenhum pode ceder parte da memória que tem
- Solução :
 - Reducir o número de processos que competem por memória
 - Passar um ou mais processos para disco e atribuir as páginas que lhes estavam atribuídas
 - Rever o grau de multiprogramação

Sistemas Operativos - 2018/2019 180

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 - 2018/2019

181

Tamanho das páginas

- Overhead estimado:

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

Espaço da tabela de páginas

Fragmentação interna

- Em que

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

Valor óptimo quando

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

Sistemas Operativos - 2018/2019

182

Programa

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

Sistemas Operativos - 2018/2019

183

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	5 MB/sec
40x CD-ROM	6 MB/sec
Fast Ethernet	12.5 MB/sec
ISA bus	16.7 MB/sec
EIDE (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	300 MB/sec
PCI bus	528 MB/sec
Sun Gigaplane XB backplane	20 GB/sec

Sistemas Operativos - 2018/2019

184

GSD
Grupo de Sistemas Distribuídos

E/S mapeado em memória

The diagram shows three memory mapping schemes:

- (a) Two address spaces: A large vertical rectangle labeled "Memory" is at the top, with a small horizontal bar labeled "I/O ports" below it. An arrow points from the bottom of the "Memory" bar to the number "0" at the bottom, with "0xFFFF..." above it.
- (b) One address space: A single large vertical rectangle labeled "Memory" is shown.
- (c) Two address spaces: A large vertical rectangle labeled "Memory" is at the top, with a small horizontal bar labeled "I/O ports" below it. An arrow points from the bottom of the "Memory" bar to the number "0" at the bottom, with "0xFFFF..." above it.

- Portas
- E/S mapeado em memória
- Híbrido

Sistemas Operativos - 2018/2019 185

GSD
Grupo de Sistemas Distribuídos

E/S mapeado em memória

The diagram compares two memory-mapped I/O architectures:

(a) Shared bus: Shows a "Bus" at the bottom with three components connected: "CPU", "Memory", and "I/O". Arrows indicate connections between CPU and Memory, CPU and I/O, and Memory and I/O. A label "All addresses (memory and I/O) go here" points to the bus.

(b) Dedicated port: Shows a "Bus" at the bottom with three components connected: "CPU", "Memory", and "I/O". The "CPU" is connected to both "Memory" and "I/O" via separate lines. A label "CPU reads and writes of memory go over this high-bandwidth bus" points to the connection between CPU and Memory. Another label "This memory port is to allow I/O devices access to memory" points to the connection between Memory and I/O.

Sistemas Operativos - 2018/2019 186

GSD
Grupo de Sistemas Distribuídos

Memória de Acesso Directo (DMA)

The diagram illustrates the DMA transfer process:

1. CPU programs the DMA controller (Address, Count, Control).
2. DMA requests transfer to memory.
3. Data transferred from Drive to Main memory via a Buffer.
4. Acknowledgment (Ack) sent back to the DMA controller.

Interrupt when done is triggered when the transfer is complete.

Sistemas Operativos - 2018/2019 187

GSD
Grupo de Sistemas Distribuídos

Interrupções

The diagram illustrates interrupt handling:

1. Device is finished (e.g., Disk, Keyboard, Clock, Printer).
2. Controller issues interrupt.
3. CPU acks interrupt.

The interrupt controller manages multiple peripheral devices: Disk, Keyboard, Clock, and Printer.

Sistemas Operativos - 2018/2019 188

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 - 2018/2019

189

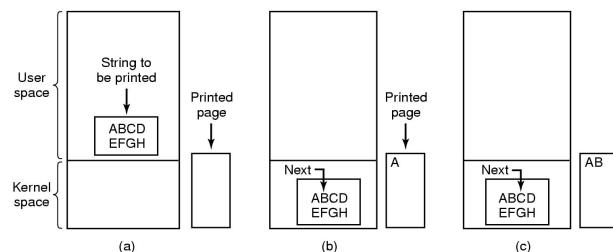
Objectivos do Software de E/S

- Transferências síncronas e assíncronas
 - chamadas bloqueantes vs. interrupções
- Armazenamento (buffering)
 - Armazenamento e cache a vários níveis
- Dispositivos partilhados vs. dedicados

Sistemas Operativos - 2018/2019

190

E/S programado



Fases na impressão de uma string

Sistemas Operativos - 2018/2019

191

E/S programado

```
copy_from_user(buffer, p, count);           /* p is the kernel bufer */
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

Sistemas Operativos - 2018/2019

192

GSD
Grupo de Sistemas Distribuídos

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) {
        unblock_user();
    } else {
        *printer_data_register = p[i];
        count = count - 1;
        i = i + 1;
    }
acknowledge_interrupt();
return_from_interrupt();

```

(a) (b)

Escrita de uma string para a impressora usando E/S por interrupções

Sistemas Operativos - 2018/2019 193

GSD
Grupo de Sistemas Distribuídos

E/S com DMA

```

copy_from_user(buffer, p, count);
set_up_DMA_controller();
scheduler();
    acknowledge_interrupt();
    unblock_user();
    return_from_interrupt();

```

(a) (b)

Operação de E/S com DMA

Sistemas Operativos - 2018/2019 194

GSD
Grupo de Sistemas Distribuídos

Níveis do software de E/S

```

graph TD
    A[User-level I/O software] --- B[Device-independent operating system software]
    B --- C[Device drivers]
    C --- D[Interrupt handlers]
    D --- E[Hardware]

```

Sistemas Operativos - 2018/2019 195

GSD
Grupo de Sistemas Distribuídos

E/S independente do dispositivo

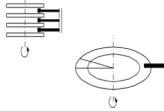
(a) Sem interface standard (b) Com interface standard

Sistemas Operativos - 2018/2019 196

GSD
Grupo de Sistemas Distribuídos

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 - 2018/2019 197

GSD
Grupo de Sistemas Distribuídos

Escalonamento de pedidos de transferência

- FIFO
- SSTF
- Elevator
- Scan circular



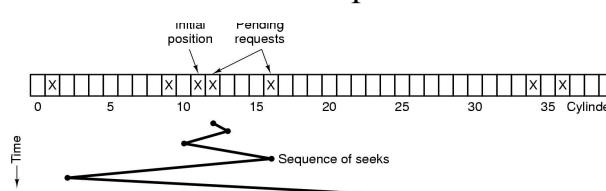
Consegue imaginar os algoritmos?

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

Sistemas Operativos - 2018/2019 198

GSD
Grupo de Sistemas Distribuídos

Escalonamento de pedidos a disco

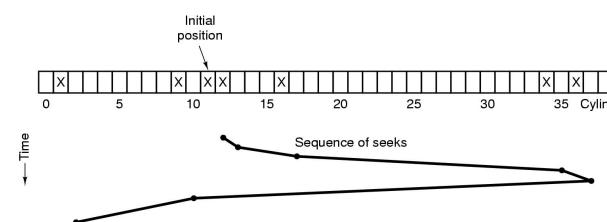


Initial position
Pending requests
Sequence of seeks
Shortest Seek First (SSF)

Sistemas Operativos - 2018/2019 199

GSD
Grupo de Sistemas Distribuídos

Escalonamento de pedidos a disco



Initial position
Sequence of seeks
Elevador

Sistemas Operativos - 2018/2019 200

GSD
Grupo de Sistemas Distribuídos

Relógios

Crystal oscillator

Counter is decremented at each pulse

Holding register is used to load the counter

Um relógio programável

Sistemas Operativos - 2018/2019

201

GSD
Grupo de Sistemas Distribuídos

Relógios

(a)

64 bits
Time of day in ticks

(b)

32 bits
Time of day in seconds

32 bits
Number of ticks in current second

(c)

32 bits
Counter in ticks

32 bits
System boot time in seconds

Três formas de manter a hora actual

Sistemas Operativos - 2018/2019

202

GSD
Grupo de Sistemas Distribuídos

Relógios

Clock header

Current time: 4200

Next signal: 3

3 → 4 → 6 → 2 → 1 | X

Simulação de vários contadores com um único relógio

Sistemas Operativos - 2018/2019

203

GSD
Grupo de Sistemas Distribuídos

Programa

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

Sistemas Operativos - 2018/2019

204

Gestão de Ficheiros

- Sistemas de ficheiros
 - Recapitação 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 - 2018/2019

205

Sistemas de ficheiros: requisitos

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

Sistemas Operativos - 2018/2019

206

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 - 2018/2019

207

Objectivos (2)

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

Sistemas Operativos - 2018/2019

208



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!**
 - Directorias
 - Podem ter milhares de entradas (eg. e-mail!)
 - Procura sequencial? Binária? B-trees?

Sistemas Operativos - 2018/2019

209



E se um disco tem uma avaria?



Sistemas Operativos - 2018/2019

211



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 - 2018/2019

210



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 - 2018/2019

212

GSD
Grupo de Sistemas Distribuidos

Sistemas RAID

(a) RAID level 0: Striping. Data is divided into strips (Strip 0, Strip 1, Strip 2, Strip 3, Strip 4, Strip 5, Strip 6, Strip 7, Strip 8, Strip 9, Strip 10, Strip 11) and distributed across two drives.

(b) RAID level 1: Mirroring. Data is replicated (Strip 0, Strip 1, Strip 2, Strip 3, Strip 4, Strip 5, Strip 6, Strip 7, Strip 8, Strip 9, Strip 10, Strip 11) across two drives.

(c) RAID level 2: Bit Interleaving. Data is striped at the bit level (Bit 1, Bit 2, Bit 3, Bit 4, Bit 5, Bit 6, Bit 7).

Sistemas Operativos - 2018/2019 213

GSD
Grupo de Sistemas Distribuidos

Sistemas RAID

(d) RAID level 3: Bit Interleaving with parity. Data is striped at the bit level (Bit 1, Bit 2, Bit 3, Bit 4, Parity) and parity is stored in a separate drive.

(e) RAID level 4: Striping with dedicated parity. Data is striped (Strip 0, Strip 1, Strip 2, Strip 3, Strip 4, Strip 5, Strip 6, Strip 7, Strip 8, Strip 9, Strip 10, Strip 11) and parity is stored in a separate drive (P0-3, P4-7, P8-11).

(f) RAID level 5: Striping with distributed parity. Data is striped (Strip 0, Strip 1, Strip 2, Strip 3, Strip 4, Strip 5, Strip 6, Strip 7, Strip 8, Strip 9, Strip 10, Strip 11, Strip 12, Strip 13, Strip 14, Strip 15, Strip 16, Strip 17, Strip 18, Strip 19) and parity is distributed across multiple drives (P0-3, P4-7, P8-11, P12-15, P16-19).

Sistemas Operativos - 2018/2019 214

GSD
Grupo de Sistemas Distribuidos

Sistemas RAID

RAID LEVEL 0 : Striped Disk Array without Fault Tolerance

Diagram showing four physical disks (A, B, C, D) connected to a single logical volume. Data is striped across the four disks.

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

Sistemas Operativos - 2018/2019 215

GSD
Grupo de Sistemas Distribuidos

Sistemas RAID

RAID LEVEL 1 : Mirroring & Duplexing

Diagram showing four pairs of physical disks (A-B, E-F, I-J, M-N) connected to a single logical volume. Each pair is mirrored (Mirroring).

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

Sistemas Operativos - 2018/2019 216

GSD
Grupo de Sistemas Distribuídos

Sistemas RAID

RAID LEVEL 5 :Independent Data Disks with Distributed Parity Blocks

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

Sistemas Operativos - 2018/2019 217

GSD
Grupo de Sistemas Distribuídos

E se há um *crash* do sistema?

?

Sistemas Operativos - 2018/2019 218

GSD
Grupo de Sistemas Distribuídos

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 - 2018/2019 219

GSD
Grupo de Sistemas Distribuídos

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 - 2018/2019 220

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 - 2018/2019

221

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 - 2018/2019

222

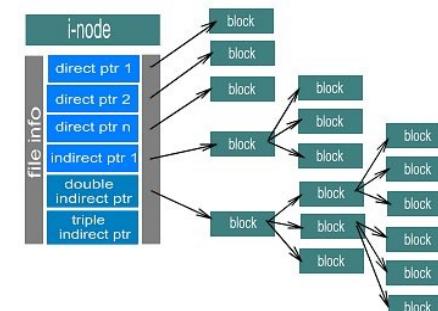
Estudo de casos

- Unix
 - Directórias + I-nodes + data blocks
 - Directórias
 - 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 - 2018/2019

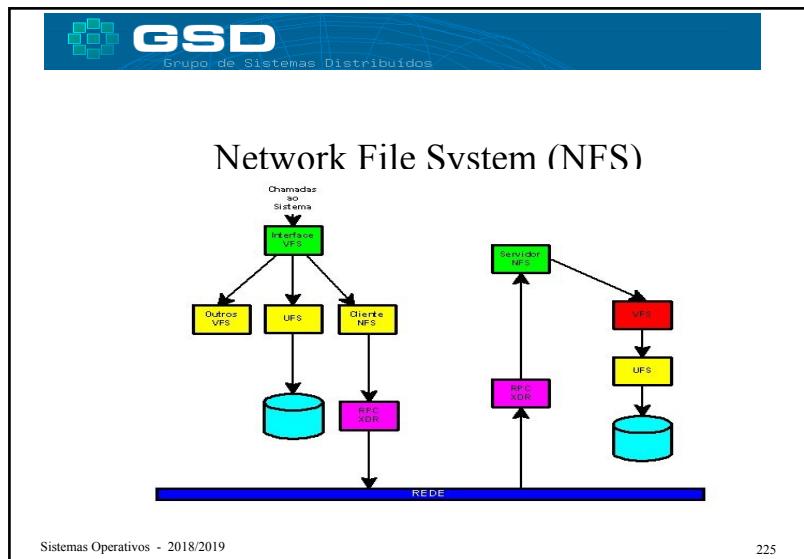
223

Estudo de casos



Sistemas Operativos - 2018/2019

224



- GSD**
Grupo de Sistemas Distribuídos
- ### E ainda...
- Extent-based file systems
 - Parallel File Systems
 - Distributed File Systems
 - Storage Area Networks
- ...
- Sistemas Operativos - 2018/2019
- 226

- GSD**
Grupo de Sistemas Distribuídos
- ### 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/) (<http://www.namesys.com/>)
- Sistemas Operativos - 2018/2019
- 227

- GSD**
Grupo de Sistemas Distribuídos
- ### 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?
- Sistemas Operativos - 2018/2019
- 228