**Trabalho de Sistemas Operacionais**

Equipe: Guilherme Daher, Lucas Medeiros e Thiago França

1)Esse algoritmo consiste em permitir que haja justiça entre os processos ao utilizar recursos de I/O. Ele distribui a largura de banda de I/O entre as requisições. O Escalonamento cria filas independentes para os processos que queria usar I/O. CFQ é um algoritmo padrão, por ser uma boa opção em sistemas onde o recurso de I/O não deve ser monopolizado por nenhum processo.

2) O CFQ não é ideal para dispositivos móveis pois sua distribuição de banda, assim como foi falado no item anterior, utilizaria grandes quantidades de recursos, fazendo com que o dispositivo móvel tivesse várias requisições em um mesmo instante de tempo e consequentemente ocorresse o travamento do aparelho.

3) Os quesitos prioritários para escalonamento de disco em dispositivos móveis são processos iterativos e os de tempo real. No primeiro ocorrem as subdivisões dos processos, de forma que a cada parte do processo que chegue, seja executado em seu devido tempo. Já o segundo é baseado em time-sharing no qual os processos são divididos em fatias de tempo e caso o tempo de execução se esgote o próximo processo será selecionado para ser executado. Tal ação simula a ideia de escalonamento em tempo real, na qual o escalonador monitora o comportamento de um processo de o ajusta dinamicamente, em tempo real.

4) Como o sistema Android é baseado em Linux, cruzaremos um paralelo entre as principais vantagens dos algoritmos de escalonamento seguindo as características mencionadas no exercício. Os algoritmo de escalonamento que iremos utilizar nessa análise são:

SIO: Corresponde a um perfil que tem como objetivo produzir o mínimo de sobrecarga ao sistema para manter baixo o nível de latência para servir as requisições.

BFQ: Bom para tarefas que não são para leitura. Como por exemplo: transferência de dados via USB, gravação de vídeos de alta definição, tem se 30% mais de rendimento.

FIFO: O primeiro que entra é o primeiro que sai, deixando de lado a resposta em tempo real, tendo em vista que seu desempenho não é desejável.

SSTF: Apresenta um desempenho bom e com a implementação adequada (evitar starvation).

SCAN: Apresenta um desempenho melhor que o FIFO e o SSTF. Adequado para pedidos que necessitam de uma resposta em “tempo real”.

C-LOOK: Semelhante ao SCAN, podendo deixar de realizar tarefas intermediárias

Deadline: Determina um tempo para os pedidos serem concluídos e prioriza os pedidos mais antigos. Não seria a melhor escolha para a atarefa, pois o algoritmo Deadline não se adequa à requisições em tempo real.

Anticipatory: Melhora o desempenho na leitura.

CFQ: Escalonador default em sistemas Linux. Divide o processo em várias filas, cada uma contendo um tempo para a realização.

Iremos à análise de cada um dos casos desejados:

Consumo de bateria: Para esse quesito, precisaremos de algoritmos mais leves, que não causem muita movimentação no disco. São eles:

1. SIO 2. Deadline 3. SSTF

Dos quais, podemos mencionar os três piores:

1. FIFO 2. C-LOOK 3. SCAN

Uso Comum: Para o uso comum queremos um escalonamento que atenda nossas necessidades na hora. Ou seja o primeiro que eu executar tem que ser o primeiro a sair. Dos algorimtos descritos, seriam os três melhores:

1. SIO 2. SCAN 3. CFQ

Com isso, conclui-se que os piores para esse tipo de tarefa são:

1. FIFO 2. BFQ 3. Não podemos apontar um terceiro pior, pois a maioria dos algoritmos acima descritos

seriam cabíveis na realização de tarefas rotineiras.

Performance em jogos: Com a análise descrita, podemos concluir que os três melhores algoritmos para desempenho em jogos são:

1. SCAN 2. Anticipatory 3. CFQ

Paralelamente, podemos tirar a conclusão dos três piores:

1. FIFO 2. C-LOOK 3. SSTF

Multitarefa pesada: Para este requisito, precisaremos de um algoritmo que garanta a realização de tarefas pesadas e ao mesmo tempo apresente um bom rendimento. Podemos citar os três melhores:

1. CFQ (como divide em filas, o desempenho aumenta para tarefas pesadas) 2. SIO 3. Deadline

Podemos relacionar os piores a:

1. Anticipatory (aumenta rendimento apenas na leitura) 2. BFQ 3. FIFO

Benchmarking: Precisamos de um algoritmo que tenha um bom rendimento com leitura e se possível, escrita e que não sobrecarregue o sistema. São os três melhores:

1. Anticipatory 2. SIO 3. CFQ

Podemos selecionar os três piores:

1. FIFO 2. BFQ 3. C-LOOK

5) Cluster mínimo (512B)

F = 512

2

\* 100 = 25600 1024

= 25K → 1024

25k

= 0,024MB

Cluster máximo (64KB)

F = 64K 2

\* 100 = 3200K →

3200K 1024

= 3,125MB

6) Tamanho máximo de clusters:

EXT3 EXT4 NTFS FAT32 extFAT

Tamanho Máximo

16TB 1EB 64KB 32 KB 128KB

Tamanho Padrão

2TB 16TB 512 bytes 512 bytes 128KB

O tamanho máximo de partição suportada pelo FAT32 é de aproximadamente 4GB, sendo assim não é possível gravar um arquivo de tamanho 5GB em um pendrive que fora formatado com FAT32, pois este aceita até no máximo 4GB de gravação de dados.

7) Slackware, Suse Enterprise, RedHat, CentOS, Solaris, FreeBSD, etc.

8) Fragmentação de arquivo é quando um arquivo não é gravado sequencialmente. Por exemplo. A1, A1, A2, A3, A3, A2. Repara que o segundo A2 não está junto do 1° arquivo A2, logo ocorre um problema de fragmentação.

A1 A1 A2 A3 A3 A2

9) NTFS: Sistema de arquivos de origem no Windows, possui tamanho máximo do olume de 16TB-4KB com várias implementações, tamanho máximo do arquivo 16EB e possui Journaling. Possui muitas melhorias em relação ao FAT32, suporta partições maiores.

EXT4: Sistema de arquivos de origem no Linux. Possui tamanho máximo do volume 1EB limitado à 16TB, tamanho máximo do arquivo 16TB e também possui Journaling. É usado por padrão em praticamente todas as distribuições Linux atuais. Possui diversas melhorias em comparação ao EXT3, incluindo um novo sistema de alocação de espaço que reduz a fragmentação dos arquivos, melhoria no desempenho. Possui velocidade de transferência muito boa.

No Windows o melhor é usar NTFS e no Linux o melhor é usar o EXT4.

10) a) pwd

b) if [ -e ~/temp ]; then echo "Existe"; fi

c) if [ -e ~/temp ]; then rm -r ~/temp; fi

d) mkdir ~/temp

e) cp /bin/\* ~/temp/

f) ls -l ~/temp

g) chmod -R 764 ~/temp

h) ls ~/temp/????

i) ls ~/temp/g???

j) zip -r dados.zip ~/temp/

k) tar -czf dados.tar.gz ~/temp/\*

l) unzip dados.zip -d backup/

m) tar -xf dados.tar.gz -C backup2

n) su - root -c "grep -R ‘senha’ /"

o) su -u root -c “groupadd alunos”

p) su -u root -c “adduser jose”

q) su - root -c “useradd -G alunos jose”

11) Letra E

12) Letra E

13) Letra A