Sistemas Operacionais

Othon Oliveira

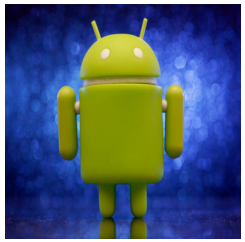
Fatec – Faculdade de Informática — PE

21 de março de 2016

1 Deadlocks

2 Starvation

Monitores – Starvation



Computadores e seus Recursos

Computadores e seus Recursos

Recursos

Os computadores têm inúmeros recursos adequados para uso de um processo por vez, podemos destacar a impressora, unidades de fitas (backup), entrada e saída nas tabelas do sistema.

Computadores e seus Recursos

Recursos

Os computadores têm inúmeros recursos adequados para uso de um processo por vez, podemos destacar a impressora, unidades de fitas (backup), entrada e saída nas tabelas do sistema.

Impasse

Se dois processos quiserem escrever simultaneamente na mesma impressora haverá um impasse.

Um ou muitos recursos ??

Acesso a muitos recursos

Para muitas aplicações, um processo necessita acesso exclusivo não somente a um recurso, mas também a vários

Um ou muitos recursos ??

Acesso a muitos recursos

Para muitas aplicações, um processo necessita acesso exclusivo não somente a um recurso, mas também a vários

Mais de um recurso para um mesmo processo

O processo A solicita primeiro permissão para usar o Scanner e é autorizado. O processo B, que é programado diferentemente, solicita primeiro permissão para usar o gravador de CD e também é autorizado. Então o processo A pede para usar o gravador de CD, mas a solicitação é negada até que o processo B o libere. Infelizmente, em vez de liberar o gravador de CD, o processo B pede para usar o Scanner, nesse ponto ambos os processos ficarão bloqueados para sempre, isso é Deadlock. Isso também pode ocorrer com computadores numa rede ?

Deadlocks em Banco de Dados

Exemplo: dois processos querem fazer *inserts*

Em sistemas de BD, um programa pode ter de bloquear o acesso a diversos registros quando estiver usando, a fim de evitar condições de disputa (race condition). Se o processo A bloquear o acesso ao registro R1, o processo B bloquear o acesso ao registro R2 e depois cada processo tentar bloquear o acesso ao registro do outro, teremos também um deadlock.

Deadlocks em Banco de Dados

Exemplo: dois processos querem fazer *inserts*

Em sistemas de BD, um programa pode ter de bloquear o acesso a diversos registros quando estiver usando, a fim de evitar condições de disputa (race condition). Se o processo A bloquear o acesso ao registro R1, o processo B bloquear o acesso ao registro R2 e depois cada processo tentar bloquear o acesso ao registro do outro, teremos também um deadlock.

Hardware e Software

Assim, deadlocks podem ocorrer tanto em recursos de hardeare com de softare.

Retirar recursos ou não ?

Recursos preemptível

Um recurso preemptível é aquele que pode ser retirado do processo proprietário sem nenhum prejuízo.

Retirar recursos ou não ?

Recursos preemptível

Um recurso preemptível é aquele que pode ser retirado do processo proprietário sem nenhum prejuízo.

Memórias

A memória é um exemplo de recurso preemptível. O processo A necessita de 32 MB de memória e é lhe cedida, o processo B quer 32 MB para imprimir algo. O computador só tem disponível 32 MB num determinado momento, a qual dos dois recursos será negada/retirada o acesso aos 32 MB ?

Sequência de recursos

Como usar recursos

Sequência de recursos

Como usar recursos

1. Requisitar o recurso

Sequência de recursos

Como usar recursos

1. Requisitar o recurso
2. Usar o recurso

Sequência de recursos

Como usar recursos

1. Requisitar o recurso
2. Usar o recurso
3. Liberar o recurso

Sequência de recursos

Como usar recursos

1. Requisitar o recurso
2. Usar o recurso
3. Liberar o recurso

Se o recurso não estiver liberado, quando requisitado, o processo será bloqueado.

Aquisição de recurso

Para alguns tipos de processo

Aquisição de recurso

Para alguns tipos de processo

Para processos de usuários cabe a eles mesmos gerir o uso de recursos

Aquisição de recurso

Para alguns tipos de processo

Para processos de usuários cabe a eles mesmos gerir o uso de recursos

Uma maneira de permitir o usuário gerir recursos é associar um semáforo a cada recurso.

Aquisição de recurso

Para alguns tipos de processo

Para processos de usuários cabe a eles mesmos gerir o uso de recursos

Uma maneira de permitir o usuário gerir recursos é associar um semáforo a cada recurso.

Esses semáforos são todos inicializados com 1

Requisitando recurso

pseudo – C

```
1
2   typedef int semaforo;
3   semaforo recurso_1;
4
5   void processo_A(void){
6     down(&recurso_1);
7     use_recurso_1();
8     libere(&recurso_1);
9   }
```

Requisitando recurso

pseudo – C

```
1
2   typedef int semaforo;
3   semaforo recurso_1;
4
5   void processo_A(void){
6     down(&recurso_1);
7     use_recurso_1();
8     libere(&recurso_1);
9   }
```

1 semáforo para 1 recurso

Uso de um semáforo para proteger um (1) recurso

Requisitando recursos

pseudo – C

```
1
2   typedef int semaforo;
3   semaforo recurso_1;
4   semaforo recurso_2;
5
6   void processo_A(void){
7     down(&recurso_1);
8     down(&recurso_2);
9     use_todos_recurso_1();
10    libere(&recurso_2);
11    libere(&recurso_1);
12  }
```

Requisitando recursos

pseudo – C

```
1
2   typedef int semaforo;
3   semaforo recurso_1;
4   semaforo recurso_2;
5
6   void processo_A(void){
7     down(&recurso_1);
8     down(&recurso_2);
9     use_todos_recurso_1();
10    libere(&recurso_2);
11    libere(&recurso_1);
12  }
```

semáforo para 2 recursos

Uso de um semáforo para proteger dois (2) recursos

Requisitando recursos

pseudo – C

```
1   typedef int semaforo;
2   semaforo recurso_1;
3   semaforo recurso_2;
4
5   void processo_A(void){
6     down(&recurso_1);
7     down(&recurso_2);
8     use_todos_recurso();
9     libere(&recurso_2);
10    libere(&recurso_1);
11  }
12  void processo_B(void){
13    down(&recurso_1);
14    down(&recurso_2);
15    use_todos_recurso();
16    libere(&recurso_2);
17    libere(&recurso_1);
```

Requisitando recursos

pseudo – C

```
1   typedef int semaforo;
2   semaforo recurso_1;
3   semaforo recurso_2;
4
5   void processo_A(void){
6     down(&recurso_1);
7     down(&recurso_2);
8     use_todos_recurso();
9     libere(&recurso_2);
10    libere(&recurso_1);
11  }
12  void processo_B(void){
13    down(&recurso_1);
14    down(&recurso_2);
15    use_todos_recurso();
16    libere(&recurso_2);
17    libere(&recurso_1);
```

Qual dos dois códigos está certo

O anterior ou este ?

```
1  typedef int semaforo;
2  semaforo recurso_1;
3  semaforo recurso_2;
4
5  void processo_A(void){
6    down(&recurso_1);
7    down(&recurso_2);
8    use_todos_recurso();
9    libere(&recurso_2);
10   libere(&recurso_1);
11 }
12 void processo_B(void){
13   down(&recurso_2);
14   down(&recurso_1);
15   use_todos_recurso();
16   libere(&recurso_1);
17   libere(&recurso_2);
```

Qual dos dois códigos está certo

O anterior ou este ?

```
1  typedef int semaforo;
2  semaforo recurso_1;
3  semaforo recurso_2;
4
5  void processo_A(void){
6    down(&recurso_1);
7    down(&recurso_2);
8    use_todos_recurso();
9    libere(&recurso_2);
10   libere(&recurso_1);
11 }
12 void processo_B(void){
13   down(&recurso_2);
14   down(&recurso_1);
15   use_todos_recurso();
16   libere(&recurso_1);
17   libere(&recurso_2);
```

Starvation

O que é Starvation?

Um processo com pouca prioridade pode ser que nunca seja selecionado

Starvation

O que é Starvation?

Um processo com pouca prioridade pode ser que nunca seja selecionado

Situação

Em programação concorrente um processo “morre” de “inanição” quando nunca é selecionado porque outros processos com maior prioridade o impedem de ser executado.

Starvation

O que é Starvation?

Um processo com pouca prioridade pode ser que nunca seja selecionado

Situação

Em programação concorrente um processo "morre" de "inanição" quando nunca é selecionado porque outros processos com maior prioridade o impedem de ser executado. Caso os processos rodem indefinidamente pode ocorrer "inanição", contudo com fatias de tempo para cada processo isso pode ser resolvido?

Starvation ou inanição ?

Fatia de tempo infinita

Caso um processo não tenha definição da fatia de tempo do processador, pode ocorrer que um processo com pouca prioridade nunca seja selecionado pelo escalonador para ser executado

Starvation ou inanição ?

Fatia de tempo infinita

Caso um processo não tenha definição da fatia de tempo do processador, pode ocorrer que um processo com pouca prioridade nunca seja selecionado pelo escalonador para ser executado

O jantar dos filósofos

- Há cinco filósofos ao redor de uma mesa

Starvation ou inanição ?

Fatia de tempo infinita

Caso um processo não tenha definição da fatia de tempo do processador, pode ocorrer que um processo com pouca prioridade nunca seja selecionado pelo escalonador para ser executado

O jantar dos filósofos

- Há cinco filósofos ao redor de uma mesa
- Um garfo é colocado entre cada filósofo

Starvation ou inanição ?

Fatia de tempo infinita

Caso um processo não tenha definição da fatia de tempo do processador, pode ocorrer que um processo com pouca prioridade nunca seja selecionado pelo escalonador para ser executado

O jantar dos filósofos

- Há cinco filósofos ao redor de uma mesa
- Um garfo é colocado entre cada filósofo
- O espaguete está escorregadio, o filósofo precisa de dois garfos para comer

Starvation ou inanição ?

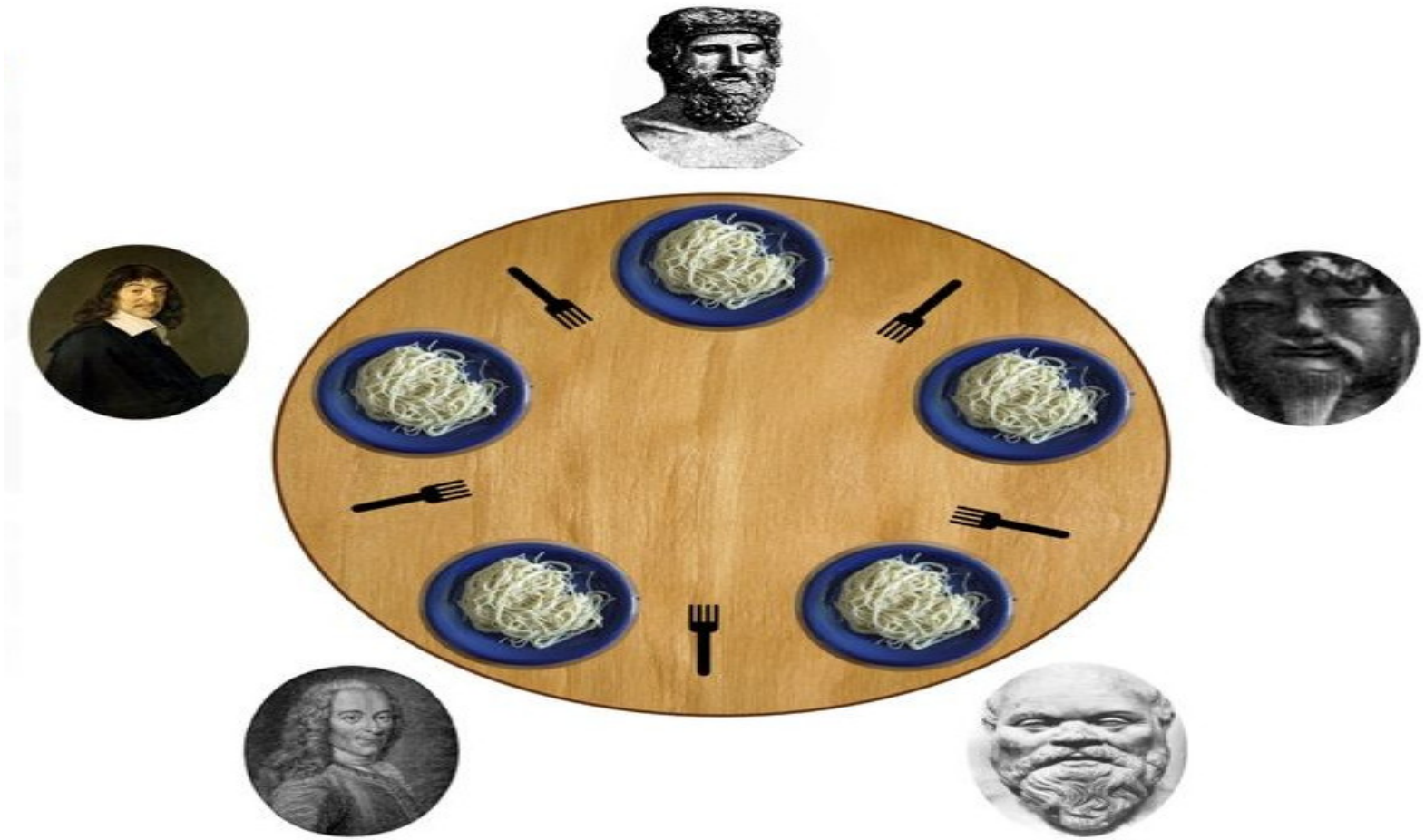
Fatia de tempo infinita

Caso um processo não tenha definição da fatia de tempo do processador, pode ocorrer que um processo com pouca prioridade nunca seja selecionado pelo escalonador para ser executado

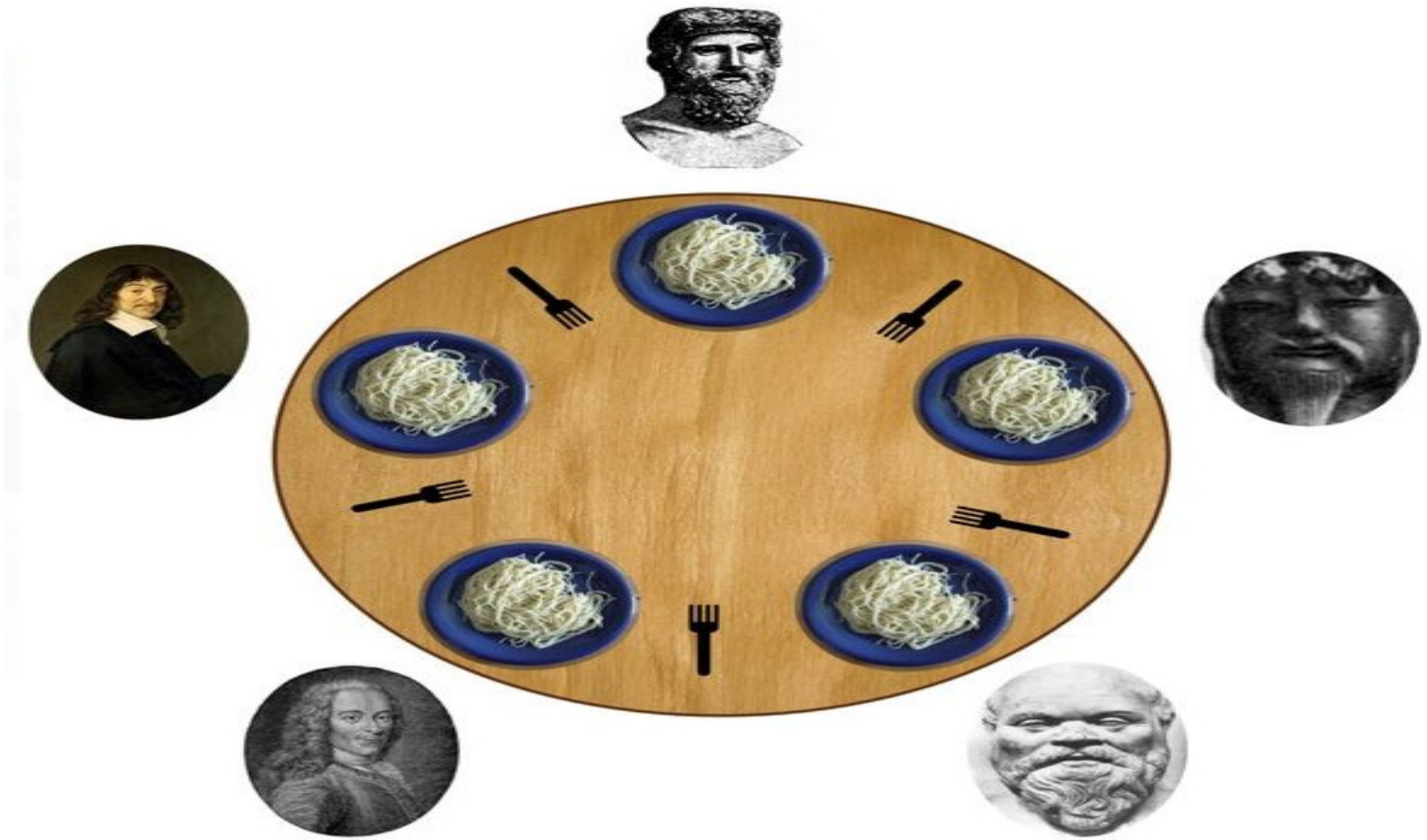
O jantar dos filósofos

- Há cinco filósofos ao redor de uma mesa
- Um garfo é colocado entre cada filósofo
- O espaguete está escorregadio, o filósofo precisa de dois garfos para comer
- Após comer o filósofo deve liberar o garfo que utilizou.

O jantar de filósofos



O jantar de filósofos



Qual a melhor solução para não ocorre Starvation ?