T2 - Escalonamento e Gerência de Memória

Vinicius Gasparini

SOP - BCC - UDESC - 2020.2

1 Exercício 1

Na Figura 1 está o diagrama de tempo de uso da CPU e do Disco. Em vermelho processos executando enquanto na fila de alta prioridade, em azul os processos em execução enquanto na fila de baixa prioridade.

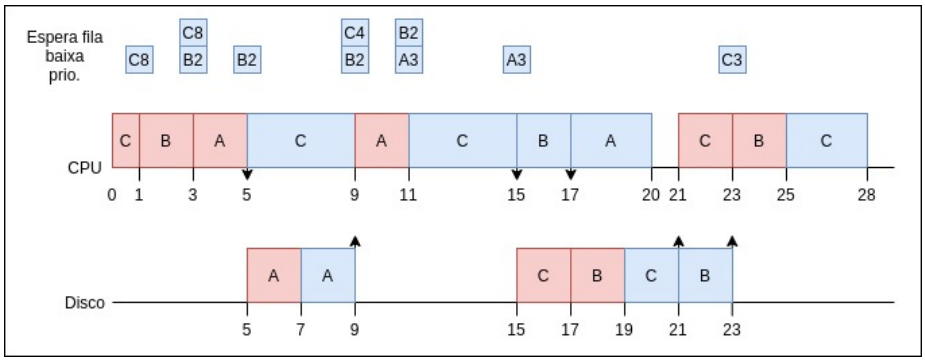

Figura 1: Diagrama de tempo de uso da CPU e Disco

O tempo médio de espera para o conjunto de processos destes 3 programas é dado pela Equação 1.

$$\begin{cases} t_a = 3 + (17 - 11) = 9 \\ t_b = 1 + (15 - 3) + (21 - 19) = 15 \\ t_c = (5 - 1) + (11 - 9) + (19 - 17) + (25 - 23) = 10 \\ \bar{t} = \frac{t_a + t_b + t_c}{3} = \frac{9+15+10}{3} = \frac{34}{3} = 11.3 \ u.t. \end{cases} \tag{1}$$

- O tempo de retorno de para o processo A é de 20 $u.t.$, para o processo B é de 25 $u.t.$ e de 28 $u.t.$ para o processo C.
- A vazão obtida para essa escala é de $\frac{3}{28} = 0.107 * 0.1 = 0.0107 \ procs/s$
- Quanto a inanição, sim é possível que ocorra inanição. Uma situação que isso poderia ocorrer seria a chegada contínua de processos antes do fim

de um *quantum* da fila de alta prioridade. Por mais que a fila de baixa prioridade preserve a ordem com a chegada de novos processos, o primeiro pode nunca ser executado se a fila de alta prioridade sempre se mantiver cheia.

2 Exercício 2

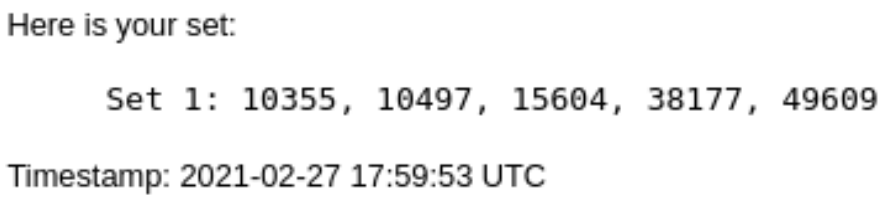

Figura 2: Conjunto de valores sorteados.

- Tamanho de página = 8KB = 8192 bytes
- EL = **10355**
  - Determinando página virtual (p): 10355/8192 = 1
  - Determinando deslocamento (d): 10355 mod 8192 = 2163
  - Bit válido da página 1 ($tabpag[1].v$): 1
  - Mapeando página física ($tagpag[1].f$): 7
  - Inicio página física 7: $7 * 8192 = 57344$
  - Endereço físico de 10355 (EF): $57344 + 2163 =$ **59507**
- EL = **10497**
  - Determinando página virtual (p): 10497/8192 = 1
  - Determinando deslocamento (d): 10497 mod 8192 = 2305
  - Bit válido da página 1 ($tabpag[1].v$): 1
  - Mapeando página física ($tagpag[1].f$): 7
  - Inicio página física 7: $7 * 8192 = 57344$
  - Endereço físico de 10355 (EF): $57344 + 2305 =$ **59649**
- EL = **15604**
  - Determinando página virtual (p): 15604/8192 = 1
  - Determinando deslocamento (d): 15604 mod 8192 = 7412
  - Bit válido da página 1 ($tabpag[1].v$): 1

- Mapeando página física ($tagpag[1].f$): 7
- Inicio página física 7: $7 * 8192 = 57344$
- Endereço físico de 10355 (EF): $57344 + 7412 =$ **64756**

- $EL =$ **38177**
  - Determinando página virtual (p): $38177/8192 = 4$
  - Determinando deslocamento (d): $38177\ mod\ 8192 = 5409$
  - Bit válido da página 1 ($tabpag[4].v$): 1
  - Mapeando página física ($tagpag[4].f$): 0
  - Inicio página física 7: $0 * 8192 = 0$
  - Endereço físico de 10355 (EF): **5409**

- $EL =$ **49609**
  - Determinando página virtual (p): $49609/8192 = 6$
  - Determinando deslocamento (d): $49609\ mod\ 8192 = 457$
  - Bit válido da página 1 ($tabpag[6].v$): 1
  - Mapeando página física ($tagpag[6].f$): 9
  - Inicio página física 7: $9 * 8192 = 73728$
  - Endereço físico de 10355 (EF): $73728 + 457 =$ **74185**

Para a resolução deste problema foi utilizado o Algoritmo 1.

3 Exercício 3

Um sistema com endereçamento virtual de 36 bits com páginas de 2KB contará com **25 bits** disponíveis para número de página e **11 bits** para registrar deslocamento. Isso pelo fato de que $2048 = 2^{11}$.

Sabe-se que o espaço ocupado por uma tabela de páginas deve ser igual ou menor ao tamanho de uma página e que cada entrada na tabela de páginas com tamanho 32 bits corresponde a 4 bytes, nossa tabela contará com $2048/4 = 512$ registros. Para endereçar 512 registros, precisamos de **9 bits** pois $2^9 = 512$. Sendo assim, a quantidade e os tamanhos dos níveis são dados por:

- Nível 1: 512 entradas = 9 bits
- Nível 2: 512 entradas = 9 bits
- Nível 3: 128 entradas = 7 bits

Com o avançar dos níveis, faz-se menos necessário a alocação completa das tabelas. Portanto, utilizar o **nível 3** para armazenar tabelas com menos entradas otimizará o uso de memória.

O endereço hexadecimal `0xbadc0ffee` tem como representação binária o número `10111010110111000000111111111101110`.

Representando esse endereço na estrutura descrita acima, temos a Tabela 1.

Nível 1	**Nível 2**	**Nível 3**	**Deslocamento**
101110101	101110000	0011111	11111101110

Tabela 1: Representação do endereço 0xbadc0ffee

4 Exercício 4

Figura 3: Conjunto de valores sorteados.

- Número de faltas de página para execução FIFO: 9
- Número de faltas de página para execução MRU: 10

5 Algoritmos

```c
typedef struct Mapping {
    int f;
    int v;
} mapping;

void traduz(mapping* tabpag, int el, int sz){
    int p = el / sz;
    int d = el % sz;
    printf("p= %d\n",p);
    printf("d= %d\n",d);
    if(tabpag[p].v){
        printf("f= %d\n",tabpag[p].f);
        printf("EF= %d\n",tabpag[p].f * sz + d);
    } else {
        printf("Page fault!");
    }
}

int main(void){
    int sz = 8192;
    int el;

    mapping tabpag[8];
    tabpag[0].f = 6; tabpag[0].v = 1;
    tabpag[1].f = 7; tabpag[1].v = 1;
    tabpag[2].f = 3; tabpag[2].v = 0;
    tabpag[3].f = 3; tabpag[3].v = 1;
    tabpag[4].f = 0; tabpag[0].v = 1;
    tabpag[5].f = 1; tabpag[5].v = 0;
    tabpag[6].f = 9; tabpag[6].v = 1;
    tabpag[7].f = 4; tabpag[7].v = 0;

    while(scanf("%d",&el) != EOF){
        traduz(tabpag, el, sz);
    }
}
```

Algoritmo 1: Tradução de endereços.