P2 se INF 1316 – Sistemas Operacionais – 2021_2

1) (2,0) Com relação ao 2º trabalho, sobre gerência de memória, descreva como você implementaria o algoritmo ótimo. Deixe claro como seria a lógica utilizada para escolher a página a sair da memória para a entrada, em seu lugar, da página recém referenciada.

   Incluiria na tabela de páginas a necessidade de acesso a cada página contabilizando o número de leituras até que ela seja necessária. A página, com maior número de leituras até ser necessária novamente, será a substituída.

2) (2,0) Seja uma memória física com 4 quadros de página (page frames), inicialmente todos vazios, e a seguinte sequência de acessos a páginas virtuais:

   a; b; c; d; c^w; e; a; b^w; d; e^w; b; a; f^w; c; e;

   onde x^w significa um acesso de escrita na página x, senão um acesso de leitura. Desconsiderando os page-faults das primeiras quatro referências a, b, c, d, conte o número de page-faults e simule a ocupação dos quadros de página da memória para os algoritmos de substituição de páginas LRU e NRU, sendo que para esse último, deve-se zerar o bit R a cada 4 referências.

   LRU
   Assume que páginas usadas recentemente, deverão ser usadas em breve novamente.
   Princípio: descartar a página que ficou sem acesso durante o periodo de tempo mais longo.
   Implementação ideal, mas impraticável.
   Lista de páginas ordenadas pelo acesso mais recente.
   É inviável pois demandaria uma manipulação da lista a cada acesso de memória.
   Alternativa (necessita de apoio de hardware): usar um contador que é incrementado por hardware a cada acesso à memória. Cada vez que uma página é acessada, atualiza este contador na entrada da TP correspondente.
   Seleciona a página com o menor contador. Periodicamente, zera o contador.
   a; b; c; d; c^w; e; a; b^w; d; e^w; b; a; f^w; c; e;

PágVirtual	Page-fault?	PágSubstituída	PágsEmMemória
a	Não		a
b	Não		b,a
c	Não		c,b,a
d	Não		d,c,b,a
c^w	Não		c^w,d,b,a
e	Sim, 1	a	e,c^w,d,b
a	Sim, 2	b	a,e,c^w,d
b^w	Sim, 3	d	b^w,a,e,c^w
d	Sim, 4	c^w	d,b^w,a,e
e^w	Não		e^w,d,b^w,a
b	Não		b^w,e^w,d,a
a	Não		a,b^w,e^w,d
f^w	Sim, 5	d	f^w,a,b^w,e^w
c	Sim, 6	e^w	c,f^w,a,b^w
e	Sim, 7	b^w	e,c, f^w,a

   Total de 7 page-faults

NRU

Cada página tem um bit de acesso (R) e de modificação (M). Os bits são setados sempre que página é acessada ou modificada. A página é carregada com permissão somente para leitura e M=0.

No primeiro acesso para escrita, o mecanismo de proteção notifica o núcleo, que seta M=1, e troca para permissão de escrita.

A cada interrupção do relógio, seta-se R=0.

Páginas são classificadas em 4 categorias:

- não referenciada, não modificada
- não referenciada, modificada
- referenciada, não modificada
- referenciada, modificada

NRU escolhe primeiro qualquer página (em ordem ascendente de categoria)

a; b; c; d; c^w; e; a; b^w; d; e^w; b; a; f^w; c; e;

PágVirtual	Page-fault?	PágSubstituída		BitR	PágsEmMemória	
a	Não			1	a	
b	Não			1,1	b,a	
c	Não			1,1,1	c,b,a	
d	Não			1,1,1,1	d,c,b,a	Zera bits R
c^w	Não			1,0,0,0	c^w,d,b,a	
e	Sim, 1	d/b/a	sai a	1,1,0,0	e,c^w,d,b	
a	Sim, 2	d/b	sai b	1,1,1,0	a,e,c^w,d	
b^w	Sim, 3		sai d	1,1,1,1	b^w,a,e,c^w	Zera bits R
d	Sim, 4	a/e	sai e	1,0,0,0	d,b^w,a,c^w	
e^w	Sim, 5		sai a	1,1,0,0	e^w,d,b^w,c^w	
b	Não			1,1,1,0	b^w,e^w,d,c^w	
a	Sim, 6		sai c^w	1,1,1,1	a,b^w,e^w,d	Zera bits R
f^w	Sim, 7	a/d	sai d	1,0,0,0	f^w,a,b^w,e^w	
c	Sim, 8		sai a	1,1,0,0	c^w,f^w,b^w,e^w	
e	Não			1,1,1,0	e^w,c^w,f^w,b^w	

Total de 8 page-faults, porém existem outras respostas válidas.

3) (3,0) Usando as chamadas ao sistema apropriadas (não usar a chamada system) construir programa escrito em linguagem C para:
   a) Criar o diretório "**so"** em seu diretório corrente; criar os subdiretórios "**a"**, "**b"** e "**c"**; Criar arquivos **".txt"** nesses subdiretórios ("**arqa.txt"**, "**arqb.txt"** e "**arqc.txt"**) e escrever textos nestes arquivos (respectivamente 80 letras 'a' em arqa, com 20 letras por linha, 160 letras 'b' em arqb, com 40 letras por linha e 320 letras 'c' em arqc, com 80 letras por linha).
   b) Exibir atributos de um dos arquivos criados (nome, permissões, tamanho, último acesso e outros à sua escolha).
   c) Buscar um dado arquivo a partir do diretório "**so"**, ler e imprimir o conteúdo do arquivo encontrado.
   d) Alterar o conteúdo do arquivo (na 3ª linha), exibindo o resultado obtido. Usar a primitiva **seek**() para realizar esta alteração.
   e) Mudar a permissão de acesso ao arquivo.

P1

```
#include <stdio.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <fcntl.h>
#include <unistd.h>
int main(void) {
   char *textoa = "aaaaaaaaaaaaaaaaaaaa";
   char *textob = "bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb";
   char *textoc =
"ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc";
   mkdi("./so", 0700);
   mkdir("./so/a", 0700);
   mkdir("./so/b", 0700);
   mkdir("./so/c", 0700);
   int fda = open("./so/a/arqa.txt", O_CREAT | O_WRONLY, 0777);
   write(fda, textoa, 21*sizeof(char));
   int fdb = open("./so/b/arqb.txt", O_CREAT | O_WRONLY, 0777);
   write(fdb, textob, 41*sizeof(char));
   int fdc = open("./so/c/arqc.txt", O_CREAT | O_WRONLY, 0777);
   write(fdc, textoc, 81*sizeof(char));
   close(fda);
   close(fdb);
   close(fdc);
   return 0;
}
```

P2

```
#include <stdlib.h>
#include <stdio.h>
#include <errno.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
int main(void) {
  struct stat fileAt;
  if (stat("./so/a/arqa.txt", &fileAt) < 0) {
    printf("File Error");
  } else {
    printf( "Permission: %d\n", fileAt.st_mode );
    printf( "Size: %lld\n", fileAt.st_size );
    printf( "Access Time: %ld\n", fileAt.st_atime );
    printf( "Device: %d\n", fileAt.st_uid );
    printf( "Permission Changed: %ld\n", fileAt.st_ctime );
  }
  return 0;
}
```

P3

```
#include <dirent.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <fcntl.h>
#include <unistd.h>
int main(void) {
  struct dirent **nome;
  int n;
  n = scandir(".", &nome, NULL, alphasort);
  if (n < 0) {
    perror("scandir");
  } else {
    while (n--) {
      if(strcmp("arqa.txt", nome[n]->d_name) == 0) {
        char read_file[1];
        int fd = open("./so/a/arqa.txt", O_RDONLY, 0777);
        read(fd, read_file, 2);
        printf("%s\n", read_file);
        exit(1);
      }
      free(nome[n]);
    }
    free(nome);
  }
}
```

P4

```
#include <dirent.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <fcntl.h>
#include <unistd.h>
int main(void) {
    struct dirent **nome;
    int n;
    n = scandir(".", &nome, NULL, alphasort);
    if (n < 0) {
        perror("scandir");
    } else {
        while (n--) {
            if(strcmp("arqa.txt", nome[n]->d_name) == 0) {
                char string[1] = "x";
                char read_file[1];
                int fd2 = open("./so/a/arqa.txt", O_RDONLY, 0777);
```

```
            int fd3 = open("./so/a/arqa.txt", O_WRONLY, 0777);
            lseek(fd3, 1, SEEK_SET);
            write(fd3, string, sizeof(char));
            read(fd2, read_file, 2);
            printf("%s\n",read_file);
            exit(1);
          }
          free(nome[n]);
        }
        puts("Aquivo não encontrado!");
        free(nome);
    }
}
```

P5

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <sys/stat.h>
int main(int argc, char **argv) {
  char mode[] = "0777";
  char buf[100] = "./so/a/arqa.txt";
  int i;
  i = strtol(mode, 0, 8);
  if (chmod (buf,i) < 0) {
    puts("Erro!");
    exit(1);
  }
  return(0);
}
```

---

4) (1,0) Dê um exemplo completo, em pseudo-código, da utilização de semáforos para tratar do problema da exclusão mútua de dois processos (pai e filho) acessando um recurso compartilhado. Não use os algoritmos apresentados no material da disciplina.

```
semaforo s = 0;
if (fork())
{       Down (s);
        RC para acesso a recurso compartilhado entre pai e filho
        Up (s);
}
else
{       Down (s);
        RC para acesso a recurso compartilhado entre pai e filho
        Up (s);
}
```

5) (2,0) Os threads A, B, C de um mesmo processo, quando executados separadamente com uso exclusivo da CPU, possuem o seguinte comportamento, onde “Bloqueia” entende-se o processo em estado de espera:

Thread A	Thread B	Thread C
Executa 4 ut	Executa 2 ut	Executa 10 ut
Bloqueia 5 ut	Bloqueia 5 ut	
Executa 4 ut	Executa 4 ut	
Bloqueia 3 ut		
Executa 5 ut		

Desenhe o diagrama de execução destas threads, supondo que elas sejam criadas nos instantes: 0 (A), 2 (B) e 14 (C), utilizando escalonamento circular com time-slice de 4. Indique o turnaround (elapsed time) do processo.