Fundamentos de Sistemas de Operação MIEI 2016/2017

Exame de Recurso - 19 de Janeiro de 2017– Duração 2h30m /1h 45m

Quem faz o exame deve resolver as questões: 1, 2, 3, 4, 6, 7, 8, 9, 10, 11, 13

Quem faz o primeiro teste, faz as questões 1 a 7 (será feita a proporção para 20)

Quem faz o segundo teste, faz as questões 8 a 13 (será feita a proporção para 20)

Nome: Número:

Avisos: *Sem consulta; a interpretação do enunciado é da responsabilidade do estudante; se necessário explicite na resposta a sua interpretação. Ver na última página informação sobre chamadas ao sistema e algumas bibliotecas.*

Questão 1 – 2,0 valores

Um sistema operativo que suporta múltiplos processos (multi-programado) e múltiplos utilizadores (multi-utilizador) tem de garantir que erros que ocorram num processo não afectam o comportamento de outros processos. Que hardware e software é que são necessários para para garantir o seguinte:

a) Impedir que os dados de um processo sejam alterados por acção de outro processo.

b) Impedir que os ficheiros de um utilizador sejam lidos ou escritos por um outro utilizador.

Questão 2 – 2,0 valores

Muitas arquitecturas hardware suportam a instrução máquina *interrupt return* (iret). Esta instrução muda o modo do CPU de modo sistema para modo utilizador

a) Em que parte do sistema operativo é que esta instrução é utilizada ?

b) A instrução *iret* é uma instrução privilegiada? Justifique.

Questão 3 – 1,5 valores

Como é sabido, quando um utilizador escreve na linha de comando do *shell* a sequência

`nome_de_ficheiro_executável > nome_de_ficheiro`

o shell

- Cria um processo *p1* para executar `nome_de_ficheiro_executável`
- Redirige o canal *stdout* de *p1* para o ficheiro *nome_de_ficheiro*

Suponha que o fragmento de código seguinte foi retirado do código da *shell*. Complete os troços em falta

```
char *args[MAXARGS];
int p, status;
// suponha que foi chamada a função makeargv e que args contém apontadores para as
// três componentes da linha e que as 3 strings estão devidamente terminadas.
...
if ( strcmp( args[1], ">" ) == 0)
    p = fork();
    if (p > 0) { // sem teste de erro









     } else {









     }
}
```

Questão 4 – 2,0 valores

Complete o seguinte programa que pretende utilizar 4 *threads* para calcular quantos elementos de uma matriz têm o valor 0. A matriz tem SIZE*SIZE posições em que cada posição pode tomar um valor entre 0 e 255; supõe-se que SIZE é múltiplo de 4.

```
#include <stdio.h>
#include <pthread.h>
#define NTHREADS 4
#define SIZE 1024
unsigned char image[SIZE][SIZE];
// espaço para eventuais declararações em falta

void *worker(void * arg) {









}
int main(int argc, char *argv[]){
  int i; pthread_id t[NTHREADS];
  // espaço para eventuais inicializações

   for (i=0; i < NTHREADS; i++)
            pthread_create( &t[i], NULL, worker, (void*)i );
   for (i=0; i < NTHREADS; i++)
            pthread_join( t[i], NULL );
//









}
```

Complete o código apresentado. Será valorizado código que minimize o número de operações de sincronização efectuadas.

Questão 5 – 2,0 valores Considere que se pretende construir um cliente e um servidor sequencial em que

- o cliente recebe do terminal o nome de um ficheiro de texto e envia-o, usando um datagrama UDP, para o servidor
- o servidor
  - envia um datagrama com o nº de bytes do ficheiro, que se supõe que existe
  - envia o ficheiro em tantos datagramas de tamanho BUFSiZE quanto o necessário
- o cliente vai mostrando no ecrã os bytes recebidos

Complete o código do servidor e do cliente

Cliente

```
#include "udp_comm.h"
#define BUFSIZE 1024
#define SERVER “localhost”
#define SERVERPORT 1234
#define CLIENTPORT 1235
char request[BUFSIZE];
char reply[BUFSIZE];
struct sockaddr_in serv;
int sock;

int main(int argc, char *argv[]){
      sock = UDP_Open(CLIENTPORT);
      // teste de erro omitido
      fgets(request, BUFSIZE, stdin);









      return 0;
}
```

Servidor

```
#include "udp_comm.h"
#define MAXW  32
#define BUFSIZE 1024
#define SERVERPORT 1234
char request [BUFSIZE];
char reply[BUFSIZE];
struct sockaddr_in cli;
int sock;

int main(int argc, char *argv[]){
      sock = UDP_Open(_________________);
      // teste de erro omitido
      while(1){
      //Receber o pedido




      // Preparar e enviar a resposta









      }

}
```

Questão 6 – 1,5 valores Faça o diagrama de estados de um processo para os dois seguintes casos:

a) é utilizado o algoritmo de escalonamento MLFQ

b) é utilizado um algoritmo de escalonamento em que cada processo tem uma prioridade fixa e distinta de todos os outros processos; em cada momento, corre o processo com maior prioridade que estiver pronto

Questão 7 – 1,0 valores

Considere um CPU com um endereço virtual com 36 bits e em que a memória central (RAM) é gerida usando páginas com 16 Kbytes (2^{14}). Qual o número de entradas da tabela de páginas de um processo? Justifique.

Questão 8 (2.0 valores) Considere um SO que usa uma MMU baseada em páginas e com um bit de validade V. O SO e o hardware suportam paginação a pedido. Explique o que acontece quando o CPU referencia uma página virtual cujo bit V está a 0.

Questão 9 (2 valores) Considere o seguinte código, em que se supõe que o ficheiro executável "*xpto*" existe na directoria corrente.

. . .

```
char *args[MAXARGS];
int p,s;
p = fork();
if (p == 0) { // sem teste de erro
      args[0] = "./xpto";   args[1]=NULL;
      execvp("./xpto", args);
}
wait(&s);
```

O sistema operativo usa a técnica *copy on write*. Explique como é que é manipulada a tabela de páginas do filho, nos três seguintes momentos

a) na criação do processo

b) na execução da instrução `args[0] = "./xpto";`

c) na execução da chamada ao sistema *execvp*

Questão 10 (2.25 valores) Considere que o periférico */dev/zeros* foi criado com os seguintes comandos executados pelo administrador de sistema.

```
mknod /dev/zeros c 298 0
chmod ugo+r /dev/zeros
```

Este dispositivo retorna tantos bytes a zero quantos os que são especificados no 3º parâmetro da chamada ao sistema *read*. Suponha que a estrutura *file_operations* associada ao periférico tem o seguinte conteúdo

```
static struct file_operations zeros_ops =
{
       .owner = THIS_MODULE,
       .open = zeros_open,
       .release = zeros_release,
       .read = zeros_read,
};
```

a) Descreva o que se passa quando é executada a chamada ao sistema.

```
int f = open( "/dev/zeros", O_RDONLY);
```

b) Descreva as acções efectuadas por `zeros_read` quando é feita a chamada ao sistema

```
read( f, buf , 20);
```

Questão 11 (2.25 valores) Considere o seguinte comando do shell

```
touch ./x.log
```

Este comando procura na directoria corrente o ficheiro *x.log* e

- se *x.log* existe, muda a data do último acesso para a data corrente
- se *x.log* não existe, cria um ficheiro vazio com este nome

Suponha que o processo que faz as chamadas tem permissões para escrever na diretoria corrente. Descreva as alterações feitas na área de meta-dados do disco que suporta o sistema de ficheiros em que está o ficheiro *x.log*.

Questão 12 (1.5 valores) Considere um disco que depois de formatado tem 2^{24} blocos e em que cada bloco tem 2048 (2^{11}) bytes; os endereços dos blocos têm 32 bits (4 bytes).

Suponha que neste disco foi colocado um sistema de ficheiros que usa uma estratégia de atribuição de espaço em disco indexada, em que os blocos atribuídos a um ficheiro são definidos por 10 endereços de blocos:

- Os endereços 0 a 7 são endereços diretos
- O endereço 8 tem um endereço de um bloco que contém endereços de blocos.
- O endereço 9 tem um endereço de um bloco que contém endereços de blocos que, por sua vez, contêm endereços de blocos.

Diga, justificando, qual é o número máximo de blocos que, usando esta estratégia de atribuição, um ficheiro pode ter? É apenas necessário indicar a expressão usado para o cálculo.

Questão 13 (1.5 valores) Suponha um sistema de ficheiros organizado da forma habitual no UNIX/Linux. Considere a execução do comando shell *rm xpto.txt*. em que supõe que *xpto.txt* existe e que o utilizador que executa o shell tem permissões para remover o ficheiro. Admita ainda que não há nenhum *hard link* a apontar para o i-node de *xpto.txt*.

Diga que blocos da área de metadados são alterados e discuta, do ponto de vista da manutenção da consistência do sistema de ficheiros, a ordem pela qual devem ser feitas as alterações nesses blocos.

_____Algumas chamadas ao sistema UNIX/Linux

int fork ( )

*int execvp (char *executable_file, char * args[ ])*

*int wait (int *status)*

int exit (int status)

int pipe (int fd[2])

int dup (int chan)

*void * mmap(void *addr, size_t len, int prot, int flags, int fd, off_t offset);*
*int fstat(int fildes, struct stat *buf);*

_____Algumas funções da biblioteca de Pthreads

Gestão de processos

*int pthread_create (pthread_t *thread, const pthread_attr_t *attr, void *(*start_routine) (void *), void *arg)*

*int pthread_join (pthread_t thread, void **retval)*

*int pthread_mutex_init (pthread_mutex_t *mutex, const pthread_mutexattr_t *attr) ou*

pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;

*int pthread_mutex_lock (pthread_mutex_t *mutex)*

*int pthread_mutex_unlock (pthread_mutex_t *mutex)*

_____ biblioteca de sockets UDP

i*nt UDP_Open (int port)*

*int UDP_FillSockAddr (struct sockaddr_in *addr, char* hostName, int port)*

*int UDP_Read (int sd, struct sockaddr_in *addr, char* buffer, int n)*

*int UDP_Write (int sd, struct sockaddr_in *addr, char* buffer, int n)*