ISEL – DEETC

Licenciatura em Engenharia Informática e de Computadores

Sistemas Operativos

2ª Época – SV 2008/2009 (09/09/2009) Duração: 2:30 H

<u>Justifique todas as suas respostas</u>.

I

1. **[1 Valor]** No contexto de uma aplicação constituída por várias tarefas (*threads*) verifica-se que cada uma tem um segmento de pilha (*stack*) próprio. Explique, por palavras suas, qual a razão da sua existência.

2. **[1,5 Valores]** Nos sistemas operativos actuais, como os sistemas Unix/Linux e Windows, verifica-se a existência de dois modos de funcionamento: o modo *kernel* e o modo *user*. Explique, por palavras suas, em que consiste cada um destes modos e qual a sua importância no correcto funcionamento do sistema operativo.

3. **[1 Valor]** Indique a razão pela qual são gerados dois ficheiros (um de extensão .dll e outro .lib), durante a construção das bibliotecas dinâmicas e as diferenças na sua utilização.

4. **[1 Valor]** Explique qual a finalidade do mecanismo de *Thread Local Storage* (TLS) da API Win32.

II

1. **[1,5 Valores]** Discuta as consequências na adopção de uma determinada dimensão do cluster a ser utilizado num sistema de gestão de ficheiros.

2. **[1,5 Valores]** Considere o seguinte código que utiliza as excepções estruturadas da API Win32 (SEH) e a gestão de memória virtual no Windows. Apresente a **<u>ordem</u>** e o **<u>resultado</u>** da execução do seguinte programa.

```
 1 INT val = 0;
 2
 3 INT filtroSEH1() {
 4   virtualAlloc( (LPVOID) 0x00100000
           , 4096, MEM_COMMIT
           , PAGE_READWRITE);
 5   return EXCEPTION_CONTINUE_EXECUTION;
 6 }
 7
 8 INT filtroSEH2() {
 9   val += 22;
10   return EXCEPTION_EXECUTE_HANDLER;
11 }
```

```
20 int _tmain(int argc, _TCHAR* argv[]){
21   int *p = 0x00100100;
22   __try {
23     __try {
24         val+=10;
25     } __finally {val+=10;}
26     __try {
27       *p = val;
28       val+=100;
29     } __ except ( filtroSEH1() ){
30       val+=200;
31     }
32   } __except ( filtroSEH2() ) {
33     printf("Handler: val=%d\n", val);
34     printf("Handler: p=%d\n", *p);
35   }
36   printf("The Fat lady is singing\n");
37 }
```

3. **[1,5 Valores]** No contexto de aplicações gráficas na Win32, um botão, também, é uma janela. Na activação (*click*) de um botão através do rato é-lhe entregue a mensagem `WM_LBUTTONUP`. No entanto, o programador associa-lhe acções através da mensagem `WM_COMMAND`. Explique, detalhadamente como é gerada a mensagem `WM_COMMAND`.

III

1. Considere uma arquitectura de suporte à gestão de memória paginada com uma estrutura de paginação a um nível, recurso à TLB e utilizando um endereçamento virtual de 32 bits e páginas de 4 KBytes. Um processo P, em execução nesse sistema, possui a tabela páginas com o conteúdo apresentado na tabela 1.

Página	Presente	Protecção	Frame
…			
14	1	R	`0x08E7F`
15	0	R	`0x00A00`
16	1	RW	`0x0043C`
17	1	RW	`0x001E0`
18	1	R	`0x0A45A`
19	1	RW	`0x00000`
…			

Tabela 1 – Tabela de páginas do processo P

a) **[2 Valores]** Para cada um dos seguintes **endereços físicos**, acedidos pelo processo, indique o respectivo **endereço virtual** justificando a sua resposta.

a.1) `0x0043C0F5`

a.2) `0x00000000`

a.3) `0x08E7FE7A`

b) **[1,5 Valores]** Indique, justificando a sua resposta, o número total de acessos à memória que o processo realiza, após iniciar a sua execução, se aceder às seguintes posições de memória virtual: `0x00011135`; `0x00011136`; `0x00011137`; `0x0000E745`; `0x00012FAE`.

2. **[1,5 Valores]** Considere uma arquitectura com gestão de memória virtual através de paginação possuindo um espaço de endereçamento virtual de 32 bits. Supondo que os processos utilizam um espaço de memória virtual próximo do seu valor máximo, compare a adopção de um esquema paginado com uma estrutura a um nível face a uma estrutura a dois níveis. Indique, também, qual a estrutura mais adequada a esta situação.

IV

1. **[3 Valores]** Considere que se pretende determinar o número total de grevistas em todos os Institutos Politécnicos do país. Assuma que tem disponível a função: `int ObterNumeroGrevistas(char *nomeEscola);` que quando invocada vai contactar a escola indicada iniciando um processo de contagem de grevistas que é síncrono e, previsivelmente, demorado. Os nomes de todas as escolas estão descritos num *array* nomeEscolas. Apresente um programa que imprima na consola o número total de grevistas em que a obtenção do número de grevistas pelas escolas seja realizada em paralelo. Indique, adicionalmente, as principais vantagens deste tipo de implementação concorrente.

```
char * nomeEscolas[] = {"ISEL", …, "ISEP"};

const int NUM_TOTAL_ESCOLAS = 133;
```

2. **[3 Valores]** Pretende-se simular a visita ao interior das grutas de Santo António existentes na reserva natural serra de Aires/Candeeiros. Por razões de segurança e capacidade de ventilação o número de visitantes é limitado a um máximo de 30 pessoas. Por outro lado, o acesso à galeria principal é feito por uma passagem muito estreita onde só cabem 3 pessoas de cada vez (tanto para entrar como para sair). Pretende-se uma estratégia que discipline o acesso à gruta fazendo com que os visitantes em excesso esperem pela sua vez à entrada. Na resolução da questão considere que os visitantes são simulados por tarefas. Implemente o código do mecanismo de sincronismo que respeite a interface `IGestorAcessoGrutas`, assim como o código da tarefa visitante.


```
class IGestorAcessoGrutas {
public:
  virtual void esperarAcederGaleria () = 0;
  virtual void sairGaleria ()  = 0;
  virtual void esperarAcessoPassagem ()= 0;
  virtual void sairPassagem () = 0;
};
```

Nuno Oliveira e Diogo Remédios

Anyone who attempts to generate random numbers by deterministic means is, of course, living in a state of sin.
John von Neumann

ANEXO

```
BOOL WINAPI CreateProcess(
  __in          LPCTSTR lpApplicationName,
  __in_out      LPTSTR lpCommandLine,
  __in          LPSECURITY_ATTRIBUTES
                              lpProcessAttributes,
  __in          LPSECURITY_ATTRIBUTES
                               lpThreadAttributes,
  __in          BOOL bInheritHandles,
  __in          DWORD dwCreationFlags,
  __in          LPVOID lpEnvironment,
  __in          LPCTSTR lpCurrentDirectory,
  __in          LPSTARTUPINFO lpStartupInfo,
  __out         LPPROCESS_INFORMATION
                              lpProcessInformation
);

VOID WINAPI ExitProcess(
  __in          UINT uExitCode
);

BOOL WINAPI GetExitCodeProcess(
  __in          HANDLE hProcess,
  __out         LPDWORD lpExitCode
);

HANDLE WINAPI CreateThread(
  __in    LPSECURITY_ATTRIBUTES lpThreadAttributes,
  __in    SIZE_T dwStackSize,
  __in    LPTHREAD_START_ROUTINE lpStartAddress,
  __in    LPVOID lpParameter,
  __in    DWORD dwCreationFlags,
  __out   LPDWORD lpThreadId
);

VOID WINAPI ExitThread(
  __in          DWORD dwExitCode
);

BOOL WINAPI GetExitCodeThread(
  __in          HANDLE hThread,
  __out         LPDWORD lpExitCode
);

DWORD WINAPI WaitForSingleObject(
  __in          HANDLE hHandle,
  __in          DWORD dwMilliseconds
);

DWORD WINAPI WaitForMultipleObjects(
  __in          DWORD nCount,
  __in          const HANDLE* lpHandles,
  __in          BOOL bWaitAll,
  __in          DWORD dwMilliseconds
);

void WINAPI InitializeCriticalSection(
  __out        LPCRITICAL_SECTION lpCriticalSection
);

void WINAPI EnterCriticalSection(
  __in_out     LPCRITICAL_SECTION lpCriticalSection
);

void WINAPI LeaveCriticalSection(
  __in_out     LPCRITICAL_SECTION lpCriticalSection
);

void WINAPI DeleteCriticalSection(
  __in_out     LPCRITICAL_SECTION lpCriticalSection
);

HANDLE WINAPI CreateMutex(
  __in     LPSECURITY_ATTRIBUTES lpMutexAttributes,
  __in     BOOL bInitialOwner,
  __in     LPCTSTR lpName
);

BOOL WINAPI ReleaseMutex(
  __in          HANDLE hMutex
);

HANDLE WINAPI CreateSemaphore(
  __in LPSECURITY_ATTRIBUTES lpSemaphoreAttributes,
  __in LONG lInitialCount,
  __in LONG lMaximumCount,
  __in LPCTSTR lpName
);

BOOL WINAPI ReleaseSemaphore(
  __in          HANDLE hSemaphore,
  __in          LONG lReleaseCount,
  __out         LPLONG lpPreviousCount
);

HANDLE WINAPI CreateEvent(
  __in     LPSECURITY_ATTRIBUTES lpEventAttributes,
  __in     BOOL bManualReset,
  __in     BOOL bInitialState,
  __in     LPCTSTR lpName
);

BOOL WINAPI SetEvent(
  __in          HANDLE hEvent
);

BOOL WINAPI ResetEvent(
  __in          HANDLE hEvent
);
```