ISEL – DEETC

Licenciatura em Engenharia Informática e de Computadores

Sistemas Operativos

1ª Época – SI 2009/2010 (29/01/2010) Duração: 2:30 H

Justifique todas as suas respostas.

I

1. Considere que coloca em execução as duas tarefas cujo código é apresentado.

a) [1 Valor] Diga o que será previsível acontecer se o sistema operativo utilizar escalonamento **não preemptivo**.

b) [1 Valor] E o que acontecerá num sistema operativo com escalonamento **preemptivo**?

```
void tarefa1 () {
  ...
  int x=0;
  while (TRUE) {
    if (x==10000) x=0;
    else ++x;
  }
  ...
}
```

```
void tarefa2 () {
  ...
  int y=0;
  while (TRUE) {
    printf("y=%d\n", y++);
    Sleep(1000);
  }
  ...
}
```

2. **[1,5 Valores]** O sistema operativo impede às aplicações a manipulação directa dos recursos de hardware da máquina. Indique como é que o sistema operativo impede que o código do utilizador realize uma instrução de leitura IN (*input byte from port*).

3. **[1,5 Valores]** Os sistemas operativos permitem a atribuição de prioridades às tarefas. Indique as razões da sua existência e na forma como são utilizados esses valores, pelo sistema operativo, em cada um dos estados do ciclo de vida das tarefas.

II

1. **[1,5 Valores]** No contexto de uma aplicação concorrente, baseada em múltiplas tarefas, que realizam adições e subtracções surgem duas hipóteses para assegurar a atomicidade das operações: **a)** recurso ao mecanismo de sincronismo `Semaphore`; **b)** utilização das funções `InterlockedAdd` e `InterlockedDecrement`. Discuta, de forma justificada, as vantagens, desvantagens e eficiência das duas alternativas.

2. **[1,5 Valores]** A API Win32 disponibiliza o mecanismo TLS. Explique em que consiste esse mecanismo e indique um exemplo em que a sua utilização é vantajosa.

3. **[1,5 Valores]** No contexto do tratamento de excepções estruturadas (SEH) da WIN32 API e na situação em que é gerada uma excepção. Indique, qual o significado de na avaliação do filtro de uma excepção ser retornado o valor EXCEPTION_CONTINUE_SEARCH e quais os possíveis comportamentos espectáveis da aplicação após essa avaliação.

III

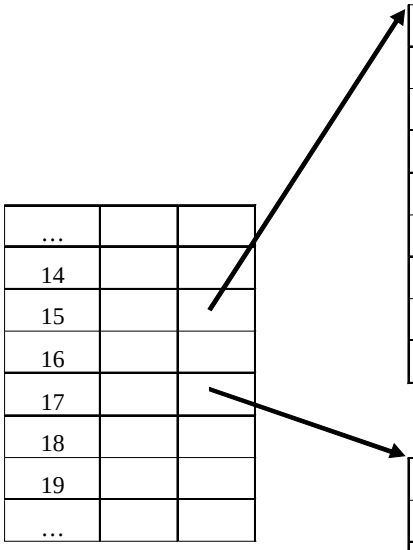
1. Considere uma arquitectura de suporte à gestão de memória paginada utilizando um endereçamento virtual de 32 bits e páginas de 4 KBytes. Um processo P, em execução nesse sistema, possui uma estrutura de suporte à paginação com a tabela de páginas apresentada na figura 1. Cada tabela ocupa a dimensão de uma página e cada entrada nas tabelas ocupa a dimensão de 4Bytes.

a) [2 Valores] Indique o endereço físico relativo aos seguintes acessos à memória virtual.

a.1) Leitura 0x04411135

a.2) Escrita 0x03C13F53

b) [1,5 Valores] Indique as acções envolvidas no acesso à posição de memória 0x0440E890 por parte de um processo.



…		
14		
15		
16		
17		
18		
19		
…		

Página	Presente	Protecção	Frame
…			
14	0	R	0x00000
15	1	R	0x00A00
16	1	RW	0x0BC00
17	1	RW	0x001E0
18	0	R	0x0A45A
19	1	RW	0x001AE
…			

Página	Presente	Protecção	Frame
…			
14	0	R	0x00000
15	1	R	0x0FA00
16	1	RW	0x0CC00
17	1	RW	0x0E1E0
18	0	R	0x0B45A
19	1	R	0x0D1AE
…			

Figura 1

2. [1,5 Valores] A comunicação entre processos pode ser um mecanismo suportado através da partilha de memória entre processos. Com base nos conhecimentos adquiridos sobre a gestão de memória virtual explique como poderá ser estabelecido um mecanismo de memória partilhada entre processos.

IV

1. **[2,5 Valores]** Uma forma de somar os valores de um *array* de reais de elevada dimensão é dividindo-o em várias partes, somar os valores dos elementos de cada uma das partes e juntar as somas parciais. Pretende-se que realize a função `double somar(double *array, int length, int numberOfThreads)`, que realiza a soma do *array* tendo em conta que deverá utilizar a forma já enunciada, ou seja a soma é realizada pelas tarefas indicadas no parâmetro `numberOfThreads`.

2. **[3 Valores]** Pretende-se simular um posto de segurança de um aeroporto. O posto de segurança é composto por três filas, uma para cada equipamento de raio-X existente. Os passageiros quando chegam ao posto de segurança escolhem a fila com menor número de pessoas e aguardam a sua vez. Uma vez em espera numa fila, apenas serão atendidos quando chegar a sua vez nessa fila, não sendo permitido trocar para outra. Numa fila, apenas será atendido um passageiro de cada vez. Este será revistado e quando terminar a sua revista sairá deixando avançar o próximo dessa fila. Considere que os passageiros são simulados por tarefas. Implemente o mecanismo de sincronismo entre as tarefas passageiros que respeite a interface IAcessoSeguranca e que garante as condições anteriormente, descritas com base no mecanismo semáforo.

```
class IAcessoSeguranca {
public:
  virtual int esperar()= 0;
  virtual void sair (int nFilaNoPosto) = 0;
};
```

Nuno Oliveira e Diogo Remédios

"The difference between school and life? In school, you're taught a lesson and then given a test. In life, you're given a test that teaches you a lesson."
-- Tom Bodett

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
);
```