ISEL – DEETC

Licenciatura em Engenharia Informática e de Computadores

Sistemas Operativos

1ª Época – SV 2008/2009 (02/07/2009) Duração: 2:30 H

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

I

1. **[1,5 Valores]** Considere uma região critica protegida por um semáforo de exclusão mútua e uma tarefa em execução dentro da zona de exclusão. Indique, para um SO multiutilizador, se pode existir preempção da tarefa enquanto esta se encontra dentro da região.

2. **[1,5 Valores]** Comente a seguinte afirmação: "O *time slice* definido nos sistemas operativos destinados aos servidores são superiores relativamente ao dos sistemas operativos dos computadores pessoais".

3. **[1,5 Valores]** Os sistemas operativos de uso generalista, como por exemplo o Windows e Unix, suportam bibliotecas de ligação dinâmicas que impõem uma complexidade acrescida na instanciação dos processos. Quais as razões da sua adopção?

II

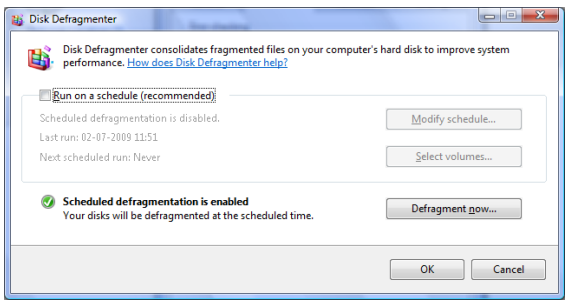
1. **[1,5 Valores]** Na gestão do sistema de ficheiros pode ser utilizado uma política de atribuição contígua. Indique, de forma justificada, as vantagens e desvantagens desta opção.

2. **[1,5 Valores]** Considere o seguinte código que utiliza as excepções estruturadas da API Win32 (SEH). Apresente a **<u>ordem</u>** e o **<u>resultado</u>** da execução do seguinte programa.

```
 1 INT val = 0;
 2
 3 INT filtroSEH1() {
 4   return EXCEPTION_CONTINUE_SEARCH;
 5 }
 6
 7 INT filtroSEH2() {
 8   val += 22;
 9   return EXCEPTION_EXECUTE_HANDLER;
10 }
```

```
20 int _tmain(int argc, _TCHAR* argv[]){
21   int *p = 0x00000100;
22   __try {
23     __try {
24         val+=10;
25     } __finally {val+=10;}
26     __try {
27       val = *p;
28       val+=100;
29     } __ except ( filtroSEH1() ){
30       val+=10;
31     }
32     __try {
33       val+=10;
34     } __except( filtroSEH2() ) {val+=11;}
35   } __except ( filtroSEH2() ) {
36     printf("Handler: val=%d\n", val);
37   }
38   printf("The Fat lady is singing\n");
39 }
```

3. **[1,5 Valores]** Na figura está representada uma janela de diálogo que permite realizar a operação de desfragmentar o disco. Discuta, fundamentando devidamente a sua resposta, o código associado ao tratamento do botão que desencadeia esta operação, indicando a forma mais correcta para o tratamento da respectiva mensagem.




```
INT_PTR CALLBACK dialogProc(HWND hDlg, UINT message
                            , WPARAM wParam, LPARAM lParam){
  switch (message){
    //...
  case WM_COMMAND:
    switch(LOWORD(wParam)){
    case IDC_BUTTON_DEFRAG:
      desfragmentarTodoODiscoRigido(); //operação síncrona
      Edit_SetText( GetDlgItem(hDlg,IDC_EDIT_RESULTADO)
         , TEXT("Operação completa, disco desfragmentado."));
      break;
      //...
    }
    //...
  }
}
```

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 tabela páginas com o conteúdo apresentado na tabela 1.

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
…			

Tabela 1 – Tabela de páginas do processo P

a) **[2 Valores]** Para cada um dos seguintes acessos à memóra indique o respectivo o endereço físico.

a.1) Leitura 0x00010135

a.2) Escrita 0x00011135

a.3) Leitura 0x00013F53

b) **[1,5 Valores]** Indique as acções envolvidas no acesso à posição de memória 0x0000E890.

2. **[1,5 Valores]** Na gestão de memória fala-se em dois tipos de fragmentação: a interna e externa. Indique, justificando as suas afirmações, quais os tipos de fragmentação existentes nos espaços de endereçamento virtual e físico num sistema baseado em paginação.

IV

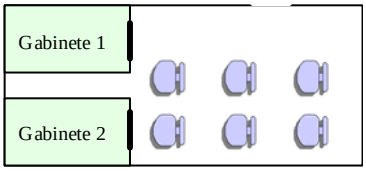
1. **[3 Valores]** Foram realizadas 100 mil entrevistas anónimas sobre o sentido de voto do próximo acto eleitoral. Os resultados dessas entrevistas estão guardados num array. Cada elemento do *array* possui o resultado de uma entrevista que pode ser um dos valores do enumerado apresentado em anexo. Apresente um programa que imprima na consola o número total de votações para cada opção de resposta (valores do enumerado *Respostas*). Este cálculo deve ser realizado através 10 tarefas concorrentes onde cada tarefa fica responsável por 10 mil posições do *array*.

```
enum Respostas { NAO_RESPONDEU=0, INDECISO,
                 BE, CDSPP, PCP, PEV, PSD, PS };

const int NUM_ENTREVISTAS = 100000;

Respostas inquerito2009[NUM_ENTREVISTAS];
```

2. [3 Valores] Pretende-se simular o atendimento dos doentes nas consultas médicas num hospital. As consultas são prestadas por dois médicos cada um no seu gabinete que partilham a mesma sala de espera com capacidade para 6 doentes. Quando os doentes chegam ao consultório verificam se tem lugar sentado e em caso afirmativo esperam para serem atendidos caso contrário desistem. Considere que os doentes são simulados por tarefas em que o seu comportamento está representado no troço de código da figura 1. As tarefas doente quando chegam ao consultório invocam o método `esperarConsulta` que, caso haja espaço na sala de espera, será bloqueante até este poder ser atendido sendo retornado o número do gabinete ao qual se deve dirigir. Na situação em que a sala de espera esteja cheia o método retorna de imediato com o valor -1.



Implemente o mecanismo de sincronismo entre os doentes que respeite a interface IGestorConsultas e que garante as condições anteriormente descritas com base no mecanismo semáforo.

```
DWORD WINAPI TarefaDoente(LPVOID arg){
  …
  int salaMedico;
  IGestorConsultas g = (IGestorConsultas*)arg;

  if( (salaMedico = g->esperarConsulta())!= -1 ){
    falarComMedico(salaMedico);
    g->sairConsulta(salaMedico);

  }else
    return -1;  //Desiste da consulta
  // vai embora
  return 0;
}
```

```
class IGestorConsultas {
public:
  virtual int esperarConsulta()= 0;
  virtual void sairConsulta (int nGab) = 0;
};
```

Figura 1 – Código da tarefa Doente e interface do gestor de sincronismo

Nuno Oliveira e Diogo Remédios

Pick battles big enough to matter, small enough to win.
-- Jonathan Kozol

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
);
```