ISEL | DEETC

Sistemas Operativos

Trabalho 3

Semestre de Inverno de 2010/2011

Autores:
31401 – Nuno Cancelo
31529 – João Sousa
32142 – Cláudia Crisóstomo
33595 – Nuno Sousa

Indicie

Enunciado

Instituto Superior de Engenharia de Lisboa

Departamento de Engenharia de Electrónica e Telecomunicações e de Computadores

Licenciatura em Engenharia Informática e de Computadores

SISTEMAS OPERATIVOS (SI-10/11)

3º Trabalho Prático – Gestão de Memória

Objectivo:

Desenvolvimento de aplicações no sistema Operativo Windows NT/2000/XP/Vista, usando a `Win32 API`; Construção de bibliotecas de ligação dinâmica (`DLL`); Tratamento de excepções estruturadas (`SEH`) da `WIN32`. Análise e desenvolvimento de aplicações `WIN32` com recurso às primitivas da API de gestão de memória e gestão de memória virtual.

Análise dos exemplos

Analise os exemplos disponíveis na página da disciplina sobre utilização de excepções SEH, DLL e Gestão de memória da `WIN32` (figura 1).

Figura 1 – Exemplos referentes às `DLL`'s, tls, excepções `SEH` e Gestão de Memória na `WIN32`

1ª Parte – Exercício prático

Considere, para efeitos da questão, que está a desenvolver um módulo a ser integrado numa aplicação e que disponibiliza um vector[1] (*array*) de elevadas dimensões (por exemplo até às dezenas de Megas) para efeitos do registo de histórico de informação (log). No entanto, este vector só é necessário ocasionalmente e as suas exigências, em termos de dimensão, variam podendo ir das centenas de bytes até ao valor máximo. **Assim, não faz sentido que a aplicação ocupe (*allocate*), inicialmente, a totalidade da dimensão**, mas sim ir atribuindo recursos físicos à medida das necessidades. O vector para o registo de informação é da exclusiva propriedade de uma tarefa podendo esta, caso necessite, iniciar um vector deste tipo. Utilizando as primitivas de gestão de memória virtual da WIN32, o tratamento de excepções estruturadas (SEH) e o mecanismo *Thread Local Storage* (TLS) para associar dados a uma tarefa específica, desenvolva uma biblioteca dinâmica (DLL) que ofereça a funcionalidade acima descrita.

As funcionalidades a disponibilizar pelo módulo são:

- Uma função para a iniciação do suporte do registo de histórico de informação, da tarefa que invoca a função, onde deve ser recebido a dimensão máxima admitida (`CreateThreadLog(…)`);
- Uma função para adicionar informação ao registo de histórico (`AppendThreadLog(…)`);
- Uma função para que permita libertar um bloco de informação mais antigo (segundo a lógica de um buffer circular), e.g. `FreeThreadLog(…, nBytes)` o que significa que o espaço dos `nBytes` mais antigos é libertado;
- Uma função para libertar todos os recursos ocupados pelo registo de histórico (`ResetThreadLog(…)`);
- Uma função para libertar o suporte do registo de histórico (`DestroyThreadLog(…)`.

Propõe-se como metodologia de realização deste trabalho a sua divisão pelas seguintes etapas:

1. Implemente as funções `CreateThreadLog`, `AppendThreadLog`, `FreeThreadLog`, `ResetThreadLog`, `DestroyThreadLog` considerando que apenas existe um vector de histórico (vector global) de forma a testar a implementação das funções no que diz respeito à reserva de memória e utilização das SEH para atribuição de memória à medida das necessidades.

[1] *Esta noção de vector assume que a disposição em memória é contígua e a indexação pode ser realizada de forma semelhante aos tradicionais Arrays das linguagens de programação.*

2. Realize um pequeno programa de teste que permita avaliar o correcto funcionamento das funções anteriores. Faça testes que levem ao armazenamento de volumes avultados de informação. Utilize o programa `VMMap`, apresentado nas aulas, para verificar a atribuição de memória.
3. **[Opcional – 4 valores]** Altere as funções desenvolvidas no ponto anterior para que o vector seja associado à tarefa que utiliza as funções. Assim cada tarefa pode possuir um vector de histórico. Neste ponto vai introduzir na implementação a utilização do mecanismo de TLS.
4. Realize um pequeno programa de teste que permita avaliar o correcto funcionamento do ponto anterior.
5. **[Opcional – 3 valores]** Organize as funções de suporte ao registo de histórico numa biblioteca dinâmica (DLL). Utilize os programas utilitários `dumpbin` ou `depends`, apresentados nas aulas, para verificar os símbolos exportados pela DLL e os símbolos importados pelo módulo executável.
6. **[Opcional – 3 valores]** Considere a possibilidade das funções `CreateThreadLog`, `AppendThreadLog`, `FreeThreadLog`, `ResetThreadLog`, `DestroyThreadLog` poderem reportar os erros através das excepções SEH.

As alíneas opcionais da metodologia sugerida, para a realização deste trabalho, possuem esta indicação juntamente com o valor que contribuem para a nota deste trabalho.

Nota:

Na resolução desta questão não pode tirar partido das funções de gestão de *heaps* disponíveis na `Win32 API`, nomeadamente, `HeapCreate(...)`, `HeapAlloc(...)` e `HeapFree(...)`.

2ª Parte – Exercícios Teóricos

1. Na figura 2 apresenta-se o código de uma função (UmaFuncao()) implementada com recurso às excepções SEH. Analise o seu código e determine o resultado devolvido, após a sua execução, descrevendo e justificando os passos da sua execução.

```
DWORD UmaFuncao() {
   DWORD dwTemp = 0;

   while (dwTemp < 10) {

      __try {
         if (dwTemp == 2)
            continue;

         if (dwTemp == 3)
            break;
      }
      __finally {
         dwTemp++;
      }

      dwTemp++;
   }

   dwTemp += 10;
   return(dwTemp);
}
```

Figura 2

2. Considere uma arquitectura com suporte à gestão de memória virtual através de paginação com uma estrutura de quatro níveis. Sabendo que a dimensão das páginas é de 4KB, cada tabela de suporte à paginação dos 4 níveis ocupa uma página e que a dimensão das entradas das tabelas é de 8 Bytes indique:
   - O esboço da organização deste sistema de gestão de memória;
   - A dimensão do espaço de endereçamento virtual;
   - O número de páginas existentes no espaço de endereçamento virtual;
   - A dimensão, mínima e máxima, ocupada pelas estruturas de gestão de memória virtuais associadas a cada processo.
3. Indique, justificando convenientemente as suas afirmações, as implicações na adopção de páginas de maior dimensão, por exemplo 8KB, na gestão de memória virtual baseada em paginação.
4. Na gestão de memória virtual com paginação que tipos de fragmentação ocorrem na espaço de endereçamento virtual e no espaço de endereçamento físico? Considere as situações em que um programa requisita várias zonas de memória (páginas) com a função VirtualAlloc e que em algumas situações não utiliza, completamente, uma página.
5. Compare a utilização das funções `VirtualAlloc` e `HeapAlloc` disponibilizados na `Win32 API`.

Entrega do trabalho de grupo

A entrega deverá ser feita até ao dia **7 de Janeiro de 2011**. Considerando a existência da concentração deste trabalho com os últimos trabalhos de outras disciplinas o trabalho poderá ser entregue, impreterivelmente, até a data de 2ª época (**Não serão aceites trabalhos após esta data**).

A entrega do trabalho **realiza-se, exclusivamente, na página da turma no Moodle**. A entrega do trabalho é constituída pelo relatório, onde lista e explica a sua solução e resultados observados, e as soluções do *Visual Studio* para que possam ser testadas na discussão.

Nas directorias das soluções do *Visual Studio* 2008 existe um ficheiro com a extensão `ncb` (que contém informação de suporte ao *Intellisense*) que deve ser eliminado de forma a reduzir a dimensão da solução a ser submetida. No caso de utilizar o *Visual Studio* 2010 deve eliminar o ficheiro com extensão `sdf` e a directoria `ipch`. Deve, igualmente, eliminar o conteúdo das directorias `Debug` e/ou `Release` da sua solução.

Todos os elementos que compõem o trabalho (relatório, código, etc.) deverão ser entregues num ficheiro comprimido do tipo *Zip* ou *Rar*. O relatório deverá ser entregue no formato *pdf*[2].

Bom trabalho,

Nuno Oliveira

[2] Poderá criar ficheiros em pdf com o seguinte utilitário grátis: http://www.primopdf.com/

Resolução

Parte I – Memoria Virtual/TLS/DLL/SEH

Durante a análise dos requesitos do enunciado deste projecto e da análise dos programas de exemplos fornecidos, a sua solução/implementação tornou-se evidente.

Esta certeza levou à esquematização de um esboço muito semelhante à versão final. O esboço contemplava todas as sugestões da metodologia, sem ter mesmo que as implementar passo a passo.

No código apresentado é evidente e compreensivel a forma como a mesma foi implementada.
A implementação do mapeamento do VirtualAlloc (e as respectivas operações e opções para reservar/alocar/desalocar e libertar) foi simples e tendo como base os exemplos fornecidos, foi um fonte de identificação de alguns erros cometidos.

A implementação da TLS foi executada em simultaneo com a implementação da DLL. Houve dificuldades na geração do modulo, mas deveu-se à "incompatibilidade" entre o utilizador e o IDE. :-D

Por fim implementou-se o retornou de erros por SEH. Este módulo do projecto causou alguns erros na interpretação da forma como retornar estas excepões, isto porque a função que procede ao retorno não é propriamente evidente.

Foram implementadas algumas excepções tipo, para testar o lançamento de excepções, seguindo as regras da Microsoft®, e tornar o programa mais personalizado.

Parte II

1. Na figura 2 apresenta-se o código de uma função (UmaFuncao()) implementada com recurso às excepções SEH. Analise o seu código e determine o resultado devolvido, após a sua execução, descrevendo e justificando os passos da sua execução..

É iniciada uma variável com o valor Zero.
No ciclo while existem três secções em execução:

1- Bloco __try:

Neste bloco testa-se o valor da variável com o valor 2. Case se confirme o valor é executada a instrução 'continue' que salta para a próxima interacção do ciclo while.
No entanto antes de proceder a este salto, o bloco __finally é executado(procedendo a um incremento da variável)
Neste bloco testa-se também o valor da variável com o valor 3. Caso se confirme o valor é executada a instrução 'break', que procede sai fora do ciclo while.
No entanto antes de proceder a este salto, o bloco __finally é executado(procedendo a um incremento da variável)

2- Bloco __finally:

Este bloco é executado sempre (no contexto desta excepções SEH), procedendo a um incremento da variável

3- incremento da variável:

incrementa sempre a variável, excepto quando é executada a instrução 'continue' e 'break'

Analizando a execução:

Interação 1:

```
dwTemp = 0
    __finally {
      dwTemp++;
    }
    dwTemp++;
dwTemp = 2
```

Interação 2:

```
dwTemp = 2
    __try {
      if (dwTemp == 2)
        continue;
    }
    __finally {
      dwTemp++;
    }
dwTemp = 3
```

Interação 3:

```
dwTemp = 3
    __try {
      if (dwTemp == 3)
        break;
    }
    __finally {
      dwTemp++;
    }
dwTemp = 4
```

fora do while

```
dwTemp += 10;
dwTemp = 14
return(dwTemp);
```

devolve 14

2. Considere uma arquitectura com suporte à gestão de memória virtual através de paginação com uma estrutura de quatro níveis. Sabendo que a dimensão das páginas é de 4KB, cada tabela de suporte à paginação dos 4 níveis ocupa uma página e que a dimensão das entradas das tabelas é de 8 Bytes indique:

- **O esboço da organização deste sistema de gestão de memória;**
- **A dimensão do espaço de endereçamento virtual;**
- **O número de páginas existentes no espaço de endereçamento virtual;**
- **A dimensão, mínima e máxima, ocupada pelas estruturas de gestão de memória virtuais associadas a cada processo.**

Informação do Enunciado:

- estrutura a 4 níveis
- Tamanho de cada página 4KB (2^2*2^{10} bytes)
- Cada tabela de suporte ocupa uma página(4 tabelas = 4 páginas = $2^2*2^2*2^{10}=2^{14}$)
- Dimensão das entradas das entradas 2^3 bytes.

Deduções:

→ o offset são 12 bits (2^2*2^{12} bytes)

→ $\frac{2^2*2^{10}}{2^3}=2^9$ entradas

→ 512 = 9 bits
→ 4 tabelas * 9 bits = 36 bits
→ endereço Virtual 4 * 9 bits + 12 bits = 48 bits
→ espaço de endereçamento 2^{48} bytes

"O esboço da organização deste sistema de gestão de memória"
Verificar Anexo A.

"A dimensão do espaço de endereçamento virtual"
→ 4 tabelas * 9 bits = 36 bits
→ endereço Virtual 4 * 9 bits + 12 bits = 48 bits
→ espaço de endereçamento 2^{48} bytes

"O número de páginas existentes no espaço de endereçamento virtual"
→ 4 tabelas * 9 bits = 36 bits
→ 2^{36} páginas

"A dimensão, mínima e máxima, ocupada pelas estruturas de gestão de memória virtuais associadas a cada processo."
→ Cada tabela ocupa uma página
Nivel 1: 1 Tabela (página) com 2^9 entradas/cada
Nivel 2: 512 Tabelas (páginas) com 2^9 entradas/cada
Nivel 3: 512 Tabelas (páginas) com $2^9 * 2^9$ entradas/cada
Nivel 4: 512 Tabelas (Páginas) com $2^9 * 2^9 * 2^9$ entradas/cada

---

Máximo: $(1+2^9+2^9*2^9+2^9*2^9*2^9)$ páginas * 4KB
por outro lado, uma vez que o tamanho das tabelas são iguais ao tamanho da pagina, podemos chegar à seguinte fórmula para obtenção do resultado:

$$\sum_{i=0}^{i=n-1} \left(2^{(i*p)}\right)$$

em que ***n*** é o numero de níveis e ***p*** o número de bits de indexação da páginas. Neste caso n=4 e p=9.
Minimo: 4 páginas * 4KB

3. Indique, justificando convenientemente as suas afirmações, as implicações na adopção de páginas de maior dimensão, por exemplo 8KB, na gestão de memória virtual baseada em paginação.

Com páginas de maior dimensão o numero de páginas diminui, diminuindo com ela o tamanho das tabelas de páginas, sendo feitas menos leituras ao disco para obtenção de páginas. Em contra partida aumenta a fragmentação interna.

4. Na gestão de memória virtual com paginação que tipos de fragmentação ocorrem na espaço de endereçamento virtual e no espaço de endereçamento físico? Considere as situações em que um programa requisita várias zonas de memória (páginas) com a função VirtualAlloc e que em algumas situações não utiliza, completamente, uma página.

Na gestão de memória virtual com paginação a fragmentação só ocorre ao nivel do espaço do endereçamento virtual. Esta é uma das razões por se implementar a gestão virtual com paginação, para evitar a fragmentação no espaço de endereçamento fisico.
Existem dois tipos de fragmentação:
Interna: Quando se aloca espaço (paginas) superior ao necessário
Externa: Quando se tem blocos de memória muito pequeno que não é possivel atribuir ou juntar para fornecer um bloco maior.

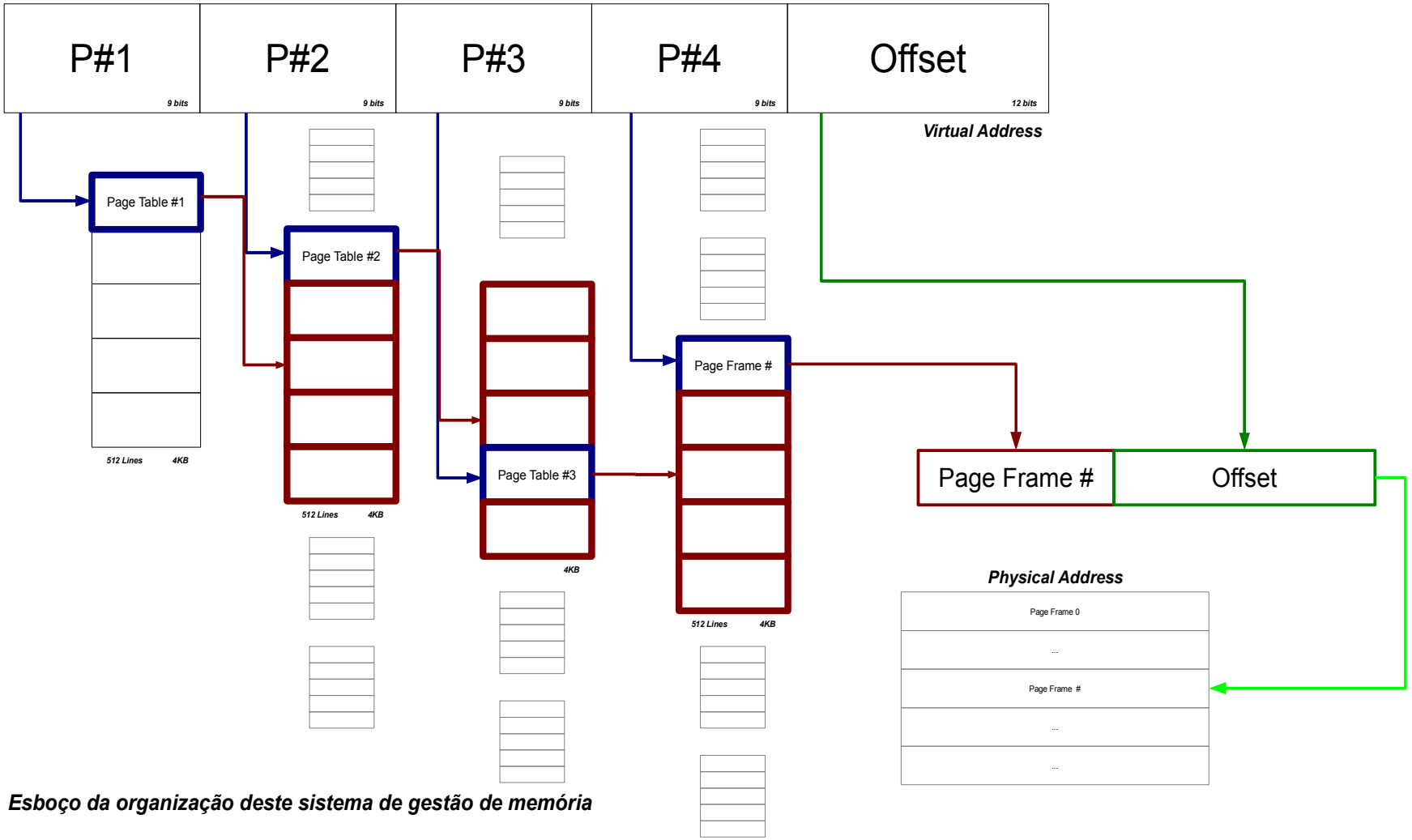
5. Compare a utilização das funções VirtualAlloc e HeapAlloc disponibilizados na Win32 API.

Quando se pretende atribuir grandes blocos de memória, em valores semelhantes/multiplos do tamanho de uma página, deve-se utilizar o VirtualAlloc que permite reservar o espaço e ir alocando à medida que se

pretenda utilizar num espaço de endereçamento contiguo.

Quando se pretende atribuir pequeno blocos de memória, de tamanho variável, deve-se utilizar o HeapAlloc

Anexo A



Esboço da organização deste sistema de gestão de memória

Conclusão

Pensamos ter alcançado os objectivos propostos.

A implementação de bibliotecas dinâmicas num “ambiente” multi-tarefa foi bastante importante para nos apercebermos da sua relevância na programação dos dias de hoje.

A utilização do mecanismo TLS e SEH, permitiu-nos desenvolver mais um pouco a programação em C++ num ambiente Windows. Apesar de trabalharmos os conceitos num Sistema Operativo Microsoft, os conceitos adquiridos permite-nos trabalhar e desenvolve-los noutros sistemas operativos, enfrentando outras realidades com os mesmos problemas/situações.

Bibliografia

Acetatos de Apoio à cadeira:

- 11 – DLL e TLS na WIN32
- 12 – SEH;
- 13 – Gestao de Memoria
- 14 - Gestao de Memoria na Win32.

Web:

- http://msdn.microsoft.com/en-us/library/
- http://www.wikipedia.com/circular-buffer

Agradecimentos

Queremos aproveitar a oportunidade para agradecer ao Prof. Nuno Oliveira pela disponíbilidade que sempre demonstrou para nos esclarecer as dúvidas que surgiram na implementação deste projecto (e não só) e toda a ajuda que nos deu para a execução do mesmo.