Licenciatura em Eng.ª Informática
Sistemas Operativos - 2º Ano - 1º Semestre

Trabalho Nº 2 - Chamadas Sistema

1. Introdução

O Sistema Operativo é constituído por um conjunto de programas que implementam uma série de funcionalidades que constituem a interface entre as aplicações do Utilizador e o Hardware de um Computador.

Para invocar as funcionalidades do Sistema Operativo, as aplicações utilizam uma interface aplicacional (API) do Sistema Operativo, designada por *System Calls* ou chamadas ao sistema.

Todas os programas que correm num sistema utilizam milhões de vezes essa API quer directamente, quer através de bibliotecas de utilitários que introduzem funcionalidade adicional e de certo modo escondem alguma da complexidade associada aos *System Calls*.

Para se entender o funcionamento dos mecanismos de base do sistema operativo, uma das primeiras abordagens possíveis é interagir com ele sem intermediários.

Com este trabalho pretende-se que os alunos adquiram esse conhecimento, através da realização de programas que irão utilizar directamente alguns dos *system calls* que fazem parte do API do sistema Linux.

2. Trabalho a realizar

2.1 Exemplo de invocação directa de um *System Call*

Vamos por a funcionar um pequeno programa escrito em GNU Assembler para imprimir uma mensagem no terminal, utilizando para o efeito uma chamada directa ao *system call* **`write()`** e a convenção de invocação dos *system calls* do Linux.

Efectue o *download* do programa **`hello.s`** disponível em:

http://netlab.ulusofona.pt/so/praticas/hello.s

Para invocar o assembler:

```
$ as hello.s –o hello.o
```

Para invocar o loader e criar o executável:

```
$ ld hello.o –o hello
```

2.2 Utilização dos *system calls* num programa em C

Utilizando a interface standard aos *system calls* fornecida pela **`libc`**, escreva um programa em C que efectue a cópia de ficheiros, segundo a sintaxe:

```
$ copy source_file dest_file
```

O programa deverá testar a existência do ficheiro de destino e imprimir uma mensagem de erro caso este já exista. Para verificar se o programa funciona correctamente deverá comparar o ficheiro de origem com o de destino e verificar se são idênticos (p.ex.

utilizando o comando **`diff`**).

Para conhecer a sintaxe de invocação das funções de biblioteca que dão acesso aos *system calls*, pode utilizar as páginas do manual do Linux através do comando:

```
$ man 2 <função>
```

Ou procurar no manual on-line disponível no site:

http://linux.die.net/

2.3 Utilização directa de *System Calls*

Quando o programa do ponto 2 estiver implementado e a funcionar correctamente, irá modificá-lo de forma a utilizar uma biblioteca de *system calls* realizada por si.

Para isso, deverá identificar todas as chamadas sistema utilizadas no programa que realizou, atribuir-lhes um nome diferente e fornecer uma implementação em Assembler para cada uma delas.

Ou seja, por exemplo, em vez de utilizar o *wrapper* do *system call* **`write()`** fornecido pela **`libc`**, irá utilizar um outro definido por si, ao qual poderá chamar p.ex. **`my_write()`**. O programa realizado no ponto anterior deverá portanto ser modificado para invocar estes novos *wrappers*.

Poderá consultar o ficheiro disponível em:

http://netlab.ulusofona.pt/so/praticas/my-hello.c

para ter uma ideia da modificação que terá de fazer ao seu ficheiro para aceder aos novos *wrappers*.

O conjunto de todos os novos *wrappers* desenvolvidos por si deverão ser agrupados num único ficheiro, que poderá por exemplo, nomear **`my_syscalls.s`**

Para criar um programa executável a partir do seu programa modificado **`new_copy.c`** deverá executar o comando:

```
$ cc new_copy.c my_syscalls.s –o new_copy
```

Deverá verificar que esta nova versão do programa de cópia de ficheiros funciona de forma idêntica à anterior.

Nota: Para escrever os seus *wrappers* de *system calls*, poderá utilizar dois Assemblers diferentes: o GNU assembler, ou o NASM (ver referências no fim deste documento). Por exemplo, o programa **`hello.s`** fornecido como exemplo está escrito utilizando o GNU assembler, mas é possível obter igualmente uma versão em NASM. No caso de dúvidas deverá consultar o professor.

3. Prazo de Entrega do Trabalho

O prazo para a entrega dos trabalhos é de duas semanas a contar da aula em que o enunciado for apresentado. Trabalhos entregues fora do prazo terão uma penalização de um ponto por cada dia de atraso.

A entrega deverá ser feita no Moodle num ficheiro zip (ou rar) contendo um relatório em PDF e as listagens dos programas realizados, obedecendo OBRIGATORIAMENTE ao seguinte formato:

`a123456-trabalho-N.zip`

4. Links e Referências

The GNU Assembler
http://tigcc.ticalc.org/doc/gnuasm.html

Introduction to UNIX Linux Assembly Programming - NASM
http://asm.sourceforge.net/intro/Assembly-Intro.html

Mecanismos da Invocação de Subrotinas na Arquitectura Intel
http://unixwiz.net/techtips/win32-callconv-asm.html

Linux System Calls Table and Definition
http://docs.cs.up.ac.za/programming/asm/derick_tut/syscalls.html
http://lxr.linux.no/#linux+v2.6.31/arch/x86/kernel/syscall_table_32.S
http://lxr.linux.no/#linux+v2.6.31/arch/x86/include/asm/unistd_32.h

"The C Programming Language, 2nd Edition (Ansi C)" de by Brian W. Kernighan, Dennis M. Ritchie, ed. Prentice Hall. http://cm.bell-labs.com/cm/cs/cbook

"*The C Book,* second edition, by Mike Banahan, Declan Brady and Mark Doran, ed. Addison Wesley
http://publications.gbdirect.co.uk/c_book

Manual de Referência (man pages) Linux on-line: http://linux.die.net/