Instituto Superior de Engenharia de Lisboa

Licenciatura em Engenharia Informática e de Computadores

Técnicas de Virtualização de Sistemas, Inverno de 2022/2023

Época Normal — Teste Parcial #1 — Teste Global, parte 1

ATENÇÃO: Responda às questões **1** e **2** num conjunto de folhas e às questões **3**, **4** e **5** noutro conjunto.

1. [6] Em Unix, o comando `sh -c <cmd>` invoca um *shell* para executar o comando `<cmd>`. Por exemplo, o comando `sh -c "ls *.c"` lista os ficheiros fonte de linguagem C presentes na diretoria corrente.

   a. [4] Implemente a função `int do_cmd(const char *cmd)`, que tira partido de um *shell* `sh` para executar o comando `cmd`. A função retorna 0 **após terminar** a execução do comando especificado ou –1 no caso de `cmd` não chegar a ser executado.

   b. [2] Descreva quais os processos criados durante a execução da seguinte chamada à função `do_cmd`, o que faz cada um deles e que tipo de resultado se espera obter no ficheiro resultante.

   ```
   do_cmd("ls *.c | grep test | wc -l > out.txt");
   ```

2. [4] Durante a tradução de **um** endereço virtual por um processador *x86-64* são consultadas quatro tabelas de páginas, pela ordem indicada, nos seguintes índices:
   - 1º nível (PML4) índice 33 (PTE com bit U/S = 1)
   - 2º nível (PDP) índice 5 (PTE com bit U/S = 0)
   - 3º nível (PD) índice 11 (PTE com bit U/S = 1)
   - 4º nível (PT) índice 8 (PTE com bit U/S = 1)

   a. [2] Sabendo que, caso esta tradução tenha sucesso, o endereço físico final resultante é 0x0000600151A8C965, que endereço virtual está a ser traduzido? Apresente os cálculos.

   b. [2] Indique se o processo de tradução indicado poderá ser concluído com sucesso quando o nível de privilégio corrente do processador é 0 e quando é 3. Justifique devidamente a sua resposta.

3. [4.5] Considere o seguinte código fonte de uma biblioteca (ficheiro com extensão `.so` ou *shared object*):

```
const int PROT = PROT_READ | PROT_WRITE;
const int MFLG = MAP_ANONYMOUS | MAP_SHARED;
void * mhalloc(size_t len) { return mmap(NULL, len, PROT, MFLG, -1, 0); }
int fhree(void * ptr, size_t len) { return munmap(ptr, len); }
```

   a. [1.5] Tendo em conta apenas o código apresentado, que alterações se pode prever que terá o espaço de endereçamento de um processo (A) ao carregar esta biblioteca com `dlopen`?

   b. [1.5] Se, no âmbito do mesmo processo (A), for depois chamada a função `mhalloc(6144)`, que alteração adicional se pode prever que terá o seu espaço de endereçamento?

   c. [1.5] Se outro processo independente (B) carregar a mesma biblioteca e também chamar a função `mhalloc(6144)`, se o ponteiro retornado pela função for igual ao retornado no processo A, os dois processos estão a partilhar uma região de memória comum? Se sim, como deverá ser chamada a função `fhree` (por A, B ou ambos) para que seja libertada a memória? Se não, em que condições poderá o bloco de memória alocado por A ser partilhado com outro processo?

(continua)

4. [2.5] Explique como se desenrola uma chamada de sistema em Linux para processadores ARM64, incluindo: como são passados os argumentos, como são devolvidos os resultados e os *exception levels* envolvidos.

5. [3] Considerando as definições globais:

```
#define SIZE 1000000
int data[SIZE] = { 2, 3, 5, 7, 11, 13, 17, 19, 23, 29 };
int info[SIZE];
```

Indique, justificando, para cada linha de código abaixo, as que podem provocar **aumento** do *resident set size* (RSS) das regiões de memória do processo com origem em secções `data` ou `bss`. Note que as linhas não são necessariamente contíguas no código fonte, podendo ocorrer alterações ao RSS entre cada linha.

a. `data[10] = 31;`
b. `printf("data[0]=%d\n", data[0]);`
c. `printf("info[0]=%d\n", info[0]);`
d. `void * mem = mmap(NULL, SIZE, PROT_WRITE, MAP_ANONYMOUS|MAP_SHARED, -1, 0);`
e. `munmap(mem);`
f. `int child_pid = fork();`

Duração: 1 hora e 15 minutos

ISEL, 17 de janeiro de 2023