Instituto Superior de Engenharia de Lisboa

Licenciatura em Engenharia Informática e de Computadores

Técnicas de Virtualização de Sistemas, Inverno de 2023/2024

Teste Parcial #1

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

1. [6] Considere o código que será compilado para o executável **denum** e, à direita, o ficheiro **in.txt**:

```
int main(int argc, char * argv[]) {
  int c;
  while ((c = fgetc(stdin)) != EOF)
    if (!(c >= '0' && c <= '9'))    // c is not a digit
      fputc(c, stdout);
  return 0;
}
```

```
ISEL-LEIC-TVS
31! T1V3S73 20?
Lisbon, PT
31-Oct-2023
```

   a. [1.5] Justifique o que é afixado no *standard output* com a execução da seguinte linha de comando:

      `denum < in.txt | grep TVS` ← *mesmo comando para as alíneas **b** e **c***

   b. [1.5] Para a linha de comando acima, quantos processos são criados e, para cada um deles, o que referem as entradas 0 e 1 das respectivas tabelas de descritores de ficheiros?

   c. [3] Escreva um programa em C que execute especificamente a linha de comandos indicada na alínea ***a***, usando diretamente a API POSIX para criar/abrir ficheiro(s), lançar os processos necessários, ligá-los da mesma forma que o *shell* e aguardar pela sua conclusão.
   NOTA: *Usar, direta ou indiretamente, uma instância de shell terá uma classificação de zero.*

2. [4] Num sistema Linux com processador *x86-64*, um processo tem o espaço de endereçamento configurado para que os seguintes endereços virtuais sejam acessíveis em *user mode* com as permissões indicadas ao lado:
   - 0x0002468ACE0468AC r__ *(Read OK , Write NOK , Execute NOK)*
   - 0x0002468ACE246321 rw_ *(Read OK , Write OK , Execute NOK)*
   - 0x0002468ACE247753 r_x *(Read OK , Write NOK , Execute OK)*

   a. [1] O processador está a operar com 4 ou 5 níveis de tradução? Justifique.

   b. [3] Quais são as permissões disponíveis em *user mode* para os seguintes endereços? Justifique.
      0x0002468ACE2468AC 0x0002468ACE247321 0x1112468ACE247753

3. [2.5] As frases abaixo são retiradas do *Architecture Reference Manual* para processadores ARM64:

   «The SVC instruction causes a Supervisor Call exception. This provides a mechanism for unprivileged software to make a system call (...).» (B1.3.1)

   «A Supervisor Call enables software executing at EL0 to make a call to (...) software executing at EL1.» (D1.3.9)

   a. [1] Sabendo que EL0 e EL1 correspondem a níveis de privilégio em processadores ARM64, indique, com base no texto, qual dos valores é o mais privilegiado e qual é o menos privilegiado.

   b. [1.5] A instrução SVC está descrita como uma excepção e **não** tem como argumento o endereço de código para onde transferir a execução. Qual é a relevância desses factos e ainda, no caso específico do Linux, como se sabe qual dos vários *system calls* possíveis se pretende executar?

4. [3.5] Considere o seguinte código fonte de uma biblioteca (ficheiro com extensão `.so` ou *shared object*) que se pretende vir a carregar usando a função `dlopen` :

```
int count = 1;
void * xalloc(size_t npages) { ++count; return mmap(NULL, npages * 4096,
                  PROT_READ | PROT_WRITE, MAP_PRIVATE | MAP_ANONYMOUS, -1, 0); }
void   xfree(void * ptr) { --count; munmap(ptr); }
```

   a. [1.5] Um processo carregou a biblioteca e só mais tarde invocou `xalloc(4)`. O que mudou no espaço de endereçamento do processo, visível em `/proc/<pid>/smaps`, com essa invocação?

   b. [2] Se o processo invocar `dlclose` num momento em que o valor de `count` é 2, o que mudará no espaço de endereçamento, visível em `/proc/<pid>/smaps`?

5. [4] O seguinte programa em C foi compilado e executado num sistema Linux com processador x86-64 :

```
 1  const char * test = "ISEL - LEIC - TVS";
 2  char u[5000];
 3  char x;
 4  char y;
 5  int main(int argc, char * argv[]) {
 6     x = 10;
 7     y = 20;
 8     getchar();
 9     test[5] = '+';
10     return 0;
11  }
```

   a. [1] Verifica-se que a execução da linha 6 e a execução da linha 7 não causam uma mudança igual (em *bytes*) no *resident set size* (`Rss`) da região de memória correspondente a `.bss`. Qual é o valor da mudança para a linha 6 e qual é o valor para a linha 7? Porque são diferentes os valores?

   b. [2] Tendo a execução ficado parada na linha 8, executou-se outra instância do mesmo programa, noutro terminal, que também pára na linha 8. Em ambos os processos (do mesmo programa), o *resident set size* (`Rss`) da região de memória correspondente à secção `.text` vale 4KiB.

      i. [1] Quanto vale a soma do *proportional set size* (`Pss`) das duas regiões `.text`? Justifique.

      ii. [1] Como está o valor de `Pss` em relação a `Rss` nas regiões correspondentes a `.bss` em cada um dos processos, ambos ainda parados no mesmo ponto. Pode apresentar o valor como um rácio (`Rss / Pss`) ou similar.

   c. [1] A execução da linha 9 resulta na terminação abrupta da execução do programa. Em código máquina, esta instrução consiste apenas numa escrita em memória com uma instrução `movb`. Que verificação falhou no CPU, permitindo detetar o acesso inválido ao índice 5 da *string* constante?

Duração: 1 hora e 15 minutos

ISEL, 31 de outubro de 2023