Instituto Superior de Engenharia de Lisboa

Licenciatura em Engenharia Informática e de Computadores

Técnicas de Virtualização de Sistemas, Inverno de 2023/2024

Exame de Época Especial — componente escrita

---

1. [3] Considere a seguinte linha de comando e, à direita, o ficheiro **in.txt**:

   ```
   cat < in.txt | grep 1 > out.txt
   ```

   ```
   01: alpha-8
   02: beta-56
   03: gamma-1

   1 continue
   2 stop
   ```

   a. [1] Justifique o que se observa no terminal e o conteúdo final de `out.txt`.

   b. [2] Para **cada um** dos processos criados para execução da linha de comando indicada, apresente o nome do executável e o que referem as entradas 0, 1 e 2 da tabela de descritores de ficheiros.

2. [2] Num sistema Linux com processador *x86-64*, sabe-se que o endereço 0x0081049500C002F9 de um processo pertence ao mapeamento da secção `.text` do executável e que 0x0081049500C012F1 pertence ao mapeamento da secção `.data`.

   a. [0.5] Quantos níveis de tradução estão a ser usados? Justifique.

   b. [1] Na tradução destes endereços, mostre em que nível se acede ao índice 6 da respetiva tabela.

   c. [0.5] O endereço 0x0081049500C012F9 pertence ao mapeamento de que secção? Justifique.

3. [1.5] No âmbito dos processadores x86-64, o que se entende por *nível de privilégio*? Qual a relevância destes níveis de privilégio para o funcionamento do *kernel* Linux?

4. [2] Considere o seguinte código fonte de um programa em C:

```
 1  char text1[4096] = {'I','S','E','L',0}; char * text2;
 2  int main() {
 3    text2 = mmap(NULL,16,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_ANONYMOUS,-1,0);
 4    strcpy(text2, "LEIC");
 5    if (fork() == 0) {
 6      char * old_text2 = text2; puts(old_text2); strcpy(text1, "DEETC");
 7      text2 = mmap(NULL,16,PROT_READ|PROT_WRITE,MAP_SHARED|MAP_ANONYMOUS,-1,0);
 8      munmap(old_text2, 16); strcpy(text2, "TVS"); puts(old_text2);
 9    } else {
10      wait(NULL); // wait for the only child
11      puts(text1); puts(text2);
12      munmap(text2, 16);
13    }
14    return 0;
15  }
```

   a. [1] Justifique qual das instruções provoca um erro que leva à terminação do respetivo processo.

   b. [1] Apresente e justifique o que é afixado no terminal por cada uma das instruções `puts`.

5. [1.5] Cada região válida do espaço de endereçamento de um processo está geralmente associada a uma área específica de disco para operar como *backing storage*. Indique, justificando, uma circunstância em que a área designada como *backing storage* para uma dada página de uma região de memória muda abruptamente.

6. [1.5] Num sistema Linux, considere um serviço que atende os processos clientes através de um único *socket* de domínio Unix do tipo *stream*. É possível atender múltiplos clientes em simultâneo? Se sim, como se distingue cada um dos clientes no serviço (por exemplo, no momento de receber/enviar dados de/para cada um dos clientes)? Se não, como se pode desenhar uma solução para suportar este requisito?

7. [1] Como é que o *systemd* determina quais os serviços que devem ser ativados no arranque do sistema?

8. [1] Um ficheiro de unidade do tipo `service` tem, na sua secção `[Unit]`, a linha `Requires=abc.socket`. Que funcionalidade do *systemd* deverá estar a ser usada no serviço `abc` e porque aparece esta diretiva `Requires` no ficheiro `.service`?

9. [1.5] Descreva sucintamente o que distingue as máquinas virtuais de sistema das máquinas virtuais de processo. Apresente um exemplo de cada caso.

10. [1.5] O **sistema operativo** Linux Alpine 3.18.0 utiliza o *kernel* Linux na versão 6.1, enquanto que o Ubuntu 20.04.6 utiliza o *kernel* 5.15. Se, sobre um sistema base Linux, executarmos simultaneamente dois **contentores** Docker, um baseado na imagem `ubuntu:20.04` e outro na imagem `alpine:3.18.0`, estes contentores irão operar sobre duas instâncias de *kernel* diferentes? Se sim, são usadas máquinas virtuais para suportar cada *kernel*? Se não, que versão única de *kernel* é usada?

11. [2] Considere a operação `docker build` guiada por um `Dockerfile`.
    a. [1] Que condições dão origem a uma nova camada (não-vazia) na imagem resultante do *build*?
    b. [1] Na reconstrução de uma imagem, que condições permitem o reaproveitamento das imagens intermédias (em *cache*) do *build* anterior?

12. [1.5] Para criar uma solução baseada em Docker Compose, foi elaborada a especificação que se apresenta ao lado (`docker-compose.yml`). Do serviço `entry` existirá sempre apenas uma instância, que executa um serviço de distribuição de carga para as instâncias existentes do serviço `webapp`. O número de instâncias de `webapp` pode ser alterado dinamicamente modificando o valor de `scale`. Identifique os dois erros existentes na especificação, indicando que problema causam e como podem ser corrigidos.

```
1   services:
2     entry:
3       image: nginx:alpine
4       networks:
5         - frontend
6       ports:
7         - "8080:80"
8       volumes:
9         - [...] # irrelevante
10
11    webapp:
12      image: tvsapp:latest
13      networks:
14        - backend
15      ports:
16        - "9090:8888"
17
18  networks:
19    frontend:
20    backend:
```

Duração: 2 horas e 15 minutos

ISEL, 17 de fevereiro de 2024