# Instituto Superior de Engenharia de Lisboa

Licenciatura em Engenharia Informática e de Computadores

Sistemas Operativos, Verão de 2020/2021

Teste Parcial #1 / Teste Parcial #2 / Exame

---

## Parte I, Duração – 1h 15m

1) [4] Considere uma MMU com dois níveis, páginas de 1KiB e um espaço de endereçamento virtual de 16MiB. Cada tabela de páginas ocupa no máximo uma página, caso em que contém 256 entradas (PTE). Metade de cada PTE contém um page frame number e a outra metade contém bits reservados e de controlo. Justifique todas as respostas, indicando sempre os cálculos realizados.

   a) [0.5] Qual a dimensão de uma PTE?

   b) [1.5] Num endereço virtual, indique quais são os bits:
      i) de offset;
      ii) para indexação no 2º nível de tradução;
      iii) para indexação no 1º nível de tradução.

   c) [1] Qual é o número de bits de um endereço físico e quanta memória permite alcançar?

   d) [1] O endereço virtual A foi traduzido para o endereço físico 0x12345, consultando a primeira entrada da tabela de tradução de primeiro nível e a PTE de segundo nível presente no endereço físico 0x45678. Qual era o endereço virtual A?

2) [4] Considere uma execução do seguinte programa e que não há outros processos do mesmo executável:

```
#define DATA_SIZE (1024*1024)
uint8_t table[DATA_SIZE] = {1}

uint8_t data[DATA_SIZE];

uint32_t sum(uint8_t *vals, size_t size) {
  uint32_t s = 0;
  for(int i=0; i < size; ++i) s += vals[i];
  return s;
}

void fill(uint8_t *vals, size_t size,
          uint8_t v) {
  for(int i=0; i < size; ++i) vals[i]= v;
}
```

```
int main() {
  // A
  fill(table + DATA_SIZE/2, DATA_SIZE/2, 255);
  uint32_t s = sum(table, DATA_SIZE/2);

  // B

  fill(data, DATA_SIZE/2, 1);
  if (fork() == 0) {
    fill(data + DATA_SIZE/2, DATA_SIZE/2, 255);
    // C
    return 0;
  }
  wait(NULL);
  // D
}
```

Responda às questões, justificando as suas respostas:

   a) [1] No ponto **A** qual o *resident set* associado à variável `data`**?**

   b) [1] A partir do ponto **B**, caso sejam roubadas páginas pertencentes ao *array* table, há dois *backing stores* possíveis para as páginas roubadas. Explique.

   c) [1] No ponto **C** quais os *resident set* partilhado e privado associados à variável `data`?

   d) [1] No ponto **D** quais os *resident set* partilhado e privado associados à variável `data`?

3) [2] Na versão de 32 bits do kernel Linux, a parte do espaço de endereçamento virtual reservada para o kernel é quase totalmente ocupada com o mapeamento linear para a memória física. No kernel de 64 bits, reserva-se metade do espaço de endereçamento virtual de kernel para esse efeito. Qual é a necessidade ou vantagem de ter este mapeamento?

4) [4] Escreva a função `void launch_n(size_t n, void (*func)(int));` que lança `n` processos filho, cada um executando `func` com um inteiro diferente entre 0 e n - 1.ma

5) [6] Considerando o seguinte programa fonte (`spy.c`), compilado para o executável `spy`, e o ficheiro `cidades.txt`:

**spy.c**

```
#include <unistd.h>
#include <fcntl.h>
#define CHUNK 1024
char data[CHUNK];
int main(int argc, char * argv[]) {
    int fd = open(argv[1], O_WRONLY, 0666);
    size_t len;
    while ((len = read(0, data, CHUNK)) != 0) {
        write(fd, data, len);
        write(1, data, len);
    }
    close(fd);
    return 0;
}
```

**cidades.txt**

**Braga**
**Porto**
**Viseu**
**Lisboa**
**Faro**

a) [2] Indique, justificando, o conteúdo do ficheiros res.txt e do standard output após a execução de:
`cat cidades.txt | grep r | spy res.txt | grep a`

b) [4] Escreva o programa C que, tal como o seguinte comando shell, executa os três programas, devidamente ligados por *pipes*:
`grep -e r cidades.txt | spy res.txt | sort`