Instituto Superior de Engenharia de Lisboa

Licenciatura em Engenharia Informática e de Computadores

Sistemas Operativos, Verão de 2020/2021

Teste Parcial #1

---

1. [5] Considere uma MMU com tradução de endereços por paginação e 2 níveis de tabelas, onde cada tabela ocupa, no máximo, uma página. Sabendo que as entradas das tabelas de página ocupam 4 bytes e que são usados 9 bits para indexar na tabela de 1º nível e 11 bits na tabela de 2º nível, indique, justificando todas as respostas:
   a. [1,5] Qual a dimensão do espaço de endereçamento virtual?
   b. [1,5] Sabendo que todos os bits das PTE são utilizados e existem 9 bits de controlo em cada PTE, qual a dimensão do espaço de endereçamento físico?
   c. [1] Sem alterar a arquitetura da MMU será possível aumentar o espaço de endereçamento virtual de modo a que fique idêntico ao espaço de endereçamento físico calculado na alínea b)?
   d. [1] Sabendo que o endereço físico da tabela de 2º nível é 0x70000, qual o endereço físico da PTE usada nessa tabela para traduzir o endereço virtual 0x3558234?

2. [3] Um processo contém uma região de memória que corresponde diretamente à secção de dados iniciados (`.data`) do respetivo ficheiro executável. Verifica-se, para essa região, que no *resident set* apenas existem páginas no estado *private clean* e que não há espaço em *swap* ocupado com páginas desta região.
   a. [1] Num momento posterior, verifica-se que a mesma região do mesmo processo passou a ter as páginas do *resident set* no estado *shared clean*, mantendo-se a não utilização de *swap*. Indique, justificando, qual pode ter sido a causa desta diferença.
   b. [1] Depois de já ter ocorrido a situação descrita em a), verifica-se, mais tarde, que há também páginas da região marcadas como *dirty*. Indique, justificando, o que terá ocorrido e se essas páginas estão *shared* ou *private*.
   c. [1] Após ter ocorrido a situação descrita em b), é possível que existam páginas da região transferidas para *swap*? Se sim, o que leva o kernel a decidir transferir páginas para *swap*? Se não, o que faz o kernel se precisar de libertar páginas físicas associadas à região?

3. [2] Nos processadores da família *x86-64*, o limite arquitetural para o endereçamento físico é de 2^{52} *bytes* (4 PiB), com as implementações correntes dos processadores efetivamente limitadas a 2^{48} *bytes* (256 TiB). No entanto, com a implementação corrente do *kernel* Linux, com a MMU a operar no modo normal de tradução de endereços com 4 níveis de tabelas, o limite efetivo de memória física, para o *kernel*, é de apenas 2^{46} bytes (64 TiB). Porquê?

4. [4] Considerando o código abaixo, indique e justifique o *output* resultante da sua execução. Considere que o processo inicial tem *pid* 1000 e que os processos criados por este têm *pids* consecutivos a partir de 1001.

```
int main() {
  printf("#1 : %d\n", getpid());
  int * ptr1 = mmap(NULL, 4096, PROT_READ|PROT_WRITE,
                                MAP_SHARED|MAP_ANONYMOUS, -1, 0);
  *ptr1 = 1111;
  pid_t pid = fork();
  printf("#2 : %d\n", getpid());

  int * ptr2 = mmap(NULL, 4096, PROT_READ|PROT_WRITE,
                                MAP_SHARED|MAP_ANONYMOUS, -1, 0);
  *ptr2 = 2222;
  if (pid == 0) {
    printf("#3 : %d | %d | %d\n", getpid(), *ptr1, *ptr2);
    *ptr1 = 3333;
    *ptr2 = 4444;
  } else {
    waitpid(pid, NULL, 0);
    printf("#4 : %d | %d | %d\n", getpid(), *ptr1, *ptr2);
  }
  return 0;
}
```

5. [6] Considere o seguinte programa fonte, compilado para o executável **oper**:

```
int main() {
   int num;
   scanf("%d", &num);
   for (int i = 0; i < 5; ++i) printf("%d\n", num + i);
   return 0;
}
```

   a. [1.5] Indique, justificando, qual o conteúdo de *output.txt* após a execução do comando *shell*:

```
echo 797 | oper | grep 8 > output.txt
```

   b. [4.5] Escreva um programa em C que execute o comando indicado na linha anterior, lançando os processos necessários, ligando-os da mesma forma que o *shell* e aguardando pela sua conclusão.

Duração: 1 hora e 15 minutos

ISEL, 8 de maio de 2021