Instituto Superior de Engenharia de Lisboa

Licenciatura em Engenharia Informática e de Computadores

Sistemas Operativos, Verão de 2020/2021

Teste Modelo

1. [4] Considere uma MMU com um espaço de endereçamento virtual de 39 bits, páginas de 8 KiBytes, PTEs com 13 bits de controlo, tabelas com a dimensão máxima de uma página e considere ainda que o espaço de endereçamento físico é o dobro do espaço de endereçamento virtual. Apresente os cálculos nas suas respostas.

    a) [1] Qual a dimensão de uma PTE?

    b) [2] Qual o número de níveis de tabelas de páginas e o número de bits de indexação associados a cada nível.

    c) [1] Considere o endereço virtual 0x123456780. Sabendo que a tabela de último nível envolvida na tradução do endereço está presente na página física com o número 0x12345 (PFN), determine o valor do endereço físico da PTE usada na tabela de último nível.

2. [4] Das páginas seguintes indique, justificando, as que são guardadas na área de *swap* em caso de necessidade de substituição.

    a) Página modificada, associada ao mapeamento SHARED de um ficheiro

    b) Página modificada, associada ao mapeamento PRIVATE de um ficheiro

    c) Página não modificada, associada ao mapeamento PRIVATE de um ficheiro

    d) Página não modificada, associada a região de dados iniciados de um executável

    e) Página associada a região de memória partilhada anónima

3. [2] Sempre que possível o Linux mapeia a totalidade da memória física de forma linear no espaço de endereçamento de kernel. Isso é possível na arquitetura x86-64, mas nem sempre na arquitetura x86-32. Explique porquê.

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

```
#include <stdio.h>
#include <unistd.h>

int main() {
    printf("from process %d before forks\n", getpid());
    fork();
    if (fork() == 0) {
          printf("after fork in process %d\n", getpid());
    }
    else {
          printf("after fork in process %d\n", getpid());
    }
    return 0;
}
```

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

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[]) {
   int num= 0;
   if (argc == 2) {
       num = atoi(argv[1]);
   }
   else {
      scanf("%d", &num);
   }
   printf("%d\n", num +1);
   return 0;
}
```

a) [2] Indique, justificando, qual o output resultante da execução do comando shell: **oper 2 | oper ; echo done**

b) [4] Escreva o programa C equivalente ao comando shell da questão anterior. Note que terá de continuar a invocar o programa **oper**

Duração: 1 hora e 15 minutos

ISEL, 1 de maio de 2021