Fundamentos de Sistemas de Operação - Teste 6/11/2010

Sem consulta. **Duração total: 2h30 horas**

Questão 1. Explique detalhadamente o significado dos argumentos passados na invocação das seguintes chamadas ao Unix e as acções efectuadas por cada uma, explicando o seu efeito sobre as estruturas internas relevantes do sistema Unix:

a1) *open*
a2) *close*
a3) *unlink*
a4) execve

Questão 2.

a) Apresente, em pseudo-código, todas as acções desencadeadas quando um programa executado em modo utilizador, invoca uma instrução para fazer a chamada ao sistema de **operação read*() sobre um pipe* no Unix**. Deve indicar quais as acções suportadas automaticamente pelo *hardware* do computador e quais as acções suportadas pelo SO e o seu efeito nas estruturas relevantes do núcleo do sistema.

b) Apresente as diferenças entre os conceitos de *pipe* e de *ficheiro normal* no sistema Unix. Na sua resposta, considere as duas perspectivas seguintes:

(i) do programador, que utiliza as chamadas ao sistema;

(ii) do núcleo do sistema de operação, no que se refere à implementação daqueles conceitos.

Questão 3. Num sistema Unix, considere a execução do seguinte programa, admitindo que, inicialmente, o ficheiro **/temp/f1** existe, mas está vazio e sem utilizadores correntes. Admita que se têm todas as permissões necessárias para aceder ao ficheiro.

```
        char buf1[2], buf2[10];
        int p, pp[2]. nw, fd, nr, r;
1.      buf1[0] = 'a';
2.      buf1[1] = 'b';
3.      r = pipe(pp);
4.      p = fork();  if (p == -1) exit(1);
5.      if (p ==0) {
6.        fd = open("/temp/f1", O_WRONLY);
7.        nw = write(fd, buf1, 2);
8.        unlink('/temp/f1');
9.        close(fd):
10.       nr = read(pp[0], buf2, 2);
11.       nr = read(pp[0], buf2, 2);
        }
12.     else {
13.       write(pp[1], buf1, 2);
14.       fd = open('/temp/f1', O_RDONLY);
15.       nr = read(fd, buf2, 10);
16.       if (nr > 0)  write(pp[1], buf2, 2);
         }
17.     exit(0);
```

a) Repetidas execuções deste programa podem conduzir a diferentes resultados? Justifique a resposta e descreva os possíveis resultados que a execução do programa pode gerar.

b) Para cada um dos cenários de execução identificados em **a)**, explique detalhadamente o efeito da execução das operações indicadas, no contexto de cada um dos processos envolvidos, para cada uma das instruções indicadas de 1 a 17:

--- nos conteúdos do ficheiro, *pipe* e directoria envolvidos
--- nos canais abertos pelos processos e valores dos cursores de caracteres (*byte offset*)
--- nos valores das variáveis: r, p, buf1, buf2, nw, nr, fd

c) Explique como modificaria o programa para que este tivesse um comportamento bem definido e determinístico em todas as suas execuções.

<u>Questão 4</u>. Utilizando chamadas ao sistema Unix, pretende-se implementar uma função, em C, a ser executada em modo utilizador

int my_put_char(char c)

que escreve o carácter c num ficheiro, cujo nome absoluto é indicado pela constante global **filename.**

A implementação da função deve satisfazer as seguintes condições:

- da 1a primeira vez que *my_put__char()* é chamada, deve abrir um canal para o ficheiro indicado, de modo a que este canal seja sempre utilizado quando for preciso escrever, no mesmo ficheiro, nas invocações seguintes de *my_put_char()*
- no retorno, esta função devolve 0 em caso de sucesso ou -1 em caso de erro .
- deve utilizar, na implementação, um único buffer de tamanho máximo BUFSIZE == 512 bytes, que guarde os caracteres que vão sendo escritos pelo processo invocador e tal que em cada chamada de *my_put_char()* deve comportar-se do seguinte modo:
  - se o buffer estiver cheio, a implementação deve pedir para escrever os 512 bytes no ficheiro, no modo sequencial do Unix (note que o buffer irá ser reutilizado por posteriores invocações de *my_put_char)*.
  - se o buffer tiver espaço livre para pelo menos um carácter, este deve ser guardado no buffer, sem gerar uma escrita física no ficheiro em disco.

a) Apresente, em C, uma implementação desta função, baseando-se em chamadas ao sistema Unix e indicando todas as variáveis (globais e locais) e estruturas de dados que utilizar.

b) Explique as vantagens da utilização da função *my_put_char,* para escrever um carácter de um ficheiro, em comparação com uma solução que invoque directamente a seguinte chamada ao sistema: write*(fd, buf, 1)*, em que se assume que fd seria o canal aberto para o ficheiro indicado.

c) Apresente o pseudo-código C de uma função auxiliar ***int flushbuffer(),*** tal que quando invocada, desencadeie as acções necessárias para que os bytes presentes correntemente no buffer sejam escritos no ficheiro, no modo sequencial do Unix (note que o buffer irá ser reutilizado por posteriores invocações de *my_put_char*). No retorno, esta função devolve 0 em caso de sucesso ou -1 em caso de erro .