Fundamentos de Sistemas de Operação - Teste 28/10/2009
Sem consulta. **Duração total: 2h30 horas**

Questão 1. Considere as seguintes chamadas ao sistema Unix: *open(), read(), pipe(), wait()*.

Indique em quais dessas chamadas se justifica a intervenção do mecanismo do sistema de operação (SO) que suporta a multiprogramação através da comutação de contextos dos processos. Justifique a resposta, para cada uma das chamadas que indique.

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 *fork()* 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, em pseudo-código, todas as acções desencadeadas (seja por *hardware,* seja pelo SO), num sistema de multiprogramação, para efecutar a comutação de contextos entre dois processos P1 e P2, no caso em que o processo P1, quando em execução, atinge o limite do seu *Time-slice*, e sendo P2 o novo processo que o SO escolherá para execução. Desenhe, no diagrama de estados dos processos, quais as transições de estados sofridas por P1 e P2 neste caso.

Questão 3. Num sistema Unix, considere a seguinte sequência de operações, num processo P1, admitindo que, inicialmente, o ficheiro **/temp/f1** existe, mas está vazio, e o ficheiro **/temp/f2** não existe. Admita que o processo tem todos as permissões necessárias para aceder aos ficheiros e que os seus canais standard 0, 1, 2 mantêm os valores herdados do processo *shell* interativo que tenha criado P1.

```
processo P1:
        char buf1[2], buf2[10];
        int nw, fd1, nr, fd2;
        ...
        buf1[0] = 'a';
        buf1[1] = 'b';

  /*sequência de instruções A */
        link('/temp/f1', '/temp/f2');
        fd1 = open('/temp/f1', O_WRONLY);
        nw = write(fd1, buf1, 2);
        unlink('/temp/f1');
        close(fd1):
/*sequência de instruções B*/
        fd2 = open('/temp/f2', O_RDONLY);
        nr = read(fd2, buf2, 10);
        close(fd2);
        unlink('/temp/f2');
        ...
```

a) Admitindo que o processo P1 executa as sequências de instruções A e B, pela ordem indicada, explique detalhadamente o efeito de cada uma das chamadas ao sistema indicadas, detalhando os feitos de cada operação:

--- nas estruturas internas ao SO, em particular nas tabelas de canais, ficheiros e i-nodes

--- nos conteúdos dos ficheiros e directorias envolvidos
--- nos canais do processo e nos valores dos cursores de caracteres (*byte offset*)
--- nos valores das variáveis do programa: buf1, buf2, nw, nr, fd1, fd2

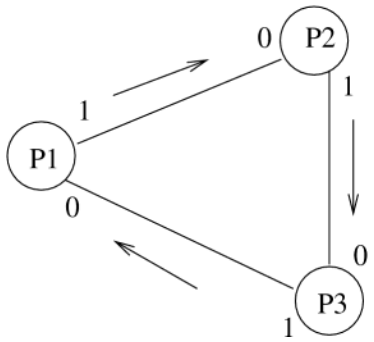
b) Explique que problemas poderiam ocorrer se a sequência de instruções A fosse executada pelo processo P1, mas a sequência B, em vez de ser executada por P1, fosse executada por outro processo P2, concorrente com P1, num sistema de multiprogramação.

<u>Questão 4</u>. Considere a seguinte configuração de processos concorrentes, no sistema Unix, onde se indicam os números dos canais standard (0 e 1) de cada processo e as suas interligações, correspondentes aos arcos da figura, realizadas através de *pipes* Unix, com o seguinte significado:

qualquer caracter escrito no canal 1 de P1 possa ser obtido através do canal 0 de P2,

qualquer caracter escrito no canal 1 de P2 possa ser obtido através do canal 0 de P3,

qualquer caracter escrito no canal 1 de P3 possa ser obtido através do canal 0 de P1,



a) Escreva um programa em C, baseado em chamadas ao sistema Unix, executado por um processo p0 inicial que crie os três processos p1, p2 e p3, e que desencadeie a realização das interligações indicadas na figura, através de *pipes*. Após as inicializações que sejam necessárias, os processos p1, p2 e p3 devem executar os programas contidos em ficheiros executáveis, cujos nomes são indicados pelas variáveis, respectivamente, P1exe, P2exe e P3exe.

b) Modifique o programa da alínea **a)**, de tal modo que os processos p1, p2, p3, criados pelo processo p0 inicial, sejam forçados a aguardar (no estado Bloqueado) até que o processo p0 lhes envie uma indicação para iniciarem a execução dos seus programas (P1exe, P2exe ou P3exe) . Utilize *pipes* pare este efeito.

c) Modifique o programa da alínea **a)** de modo a que o processo p0 inicial fique a aguardar a terminação dos processos p1, p2 e p3, e depois termine também.

d) Admita que, <u>durante a execução</u> do programa da alínea **a)**. se pretendia que, através de uma indicação enviada pelo processo p0 a cada um dos processos p1, p2, p3, o sentido das interligações indicadas na figura fosse invertido, de tal modo que o processo p1 passasse a enviar para o processo p3, o processo p3 a enviar para o processo p2 e o processo p2 a enviar para o processo p1.
Sem apresentar o programa, explique que acções seriam necessárias para esse efeito.