Instituto Superior de Engenharia de Coimbra	**Sistemas Operativos** 2013/14 DEIS – Engenharia Informática	Prova escrita Parte prática Época normal 19/6/2014

Parte I - comandos/shell

1. (10%) Utilizando apenas uma linha de comandos, e sem recorrer a ficheiros temporários, conte o número de pastas existentes da directoria /etc.

2. (10%) Utilizando apenas uma linha de comandos, e sem recorrer a ficheiros temporários, conte o número de utilizadores do sistema que têm como apelido "Santos". O nome completo dos utilizadores do sistema encontra-se no campo 5 do ficheiro /etc/passwd.

Parte II – API Unix

3. Pretende-se um sistema parecido com *chat*, mas muito simplificado. Existe um servidor responsável por receber comunicações enviadas por outros processos. Sempre que recebe uma mensagem, o servidor vai enviar essa mensagem para todos os outros processos com os quais esteja a interagir. Fundamentalmente, é isto.

Vai-se usar uma arquitectura cliente-servidor. O *chat* é controlado pelo servidor. Cada processo que envia uma mensagem (e recebe as mensagens dos outros) é um cliente. A comunicação entre ambos é feita com *named pipes*. Adianta-se que o servidor tem um *named pipe*, onde lê as mensagens dos clientes, e cada cliente tem um *named pipe* onde lê as mensagens do servidor. O *named pipe* do servidor tem um nome conhecido à partida. O nome do *named pipe* do cliente não é conhecido à partida – cada cliente tem o seu – e portanto, o cliente deve informar ao servidor o nome do seu *named pipe*.

A forma de interagir com este servidor é a seguinte: o cliente envia as mensagens que quiser a qualquer altura, sem necessidade de *login* ou anúncio prévio do tipo "olá, estou aqui". Cada mensagem deve incluir o PID, o nome do seu *named pipe*, e o texto a enviar. Também não é necessário avisar o servidor quando desliga: o servidor vai assumir que um cliente já não existe se não enviar uma mensagem a cada 30 segundos.

Pede-se que faça apenas partes bem delimitadas do servidor e do cliente. Não se querem programas completos.

a) (10%) Servidor: lançamento. Quando lançado, o servidor vai ver se já está a correr (se estiver sai logo). Não estando a correr, cria o seu pipe e entra num ciclo para lidar com os clientes.

Escreva a parte do **servidor** que é responsável por:

- Verificar se já está a correr (e sair se já estiver)
- Criar o *named pipe* do servidor.

b) (10%) Cliente: envio de uma mensagem. O cliente vai estar a ler cadeias de caracteres do teclado e envia-as para o servidor.

Escreva a parte do **cliente** que é responsável por:

- Definição da estrutura de dados correspondente a uma mensagem.
- Obter os dados para uma mensagem (cadeia de caracteres, PID, nome do *named pipe*) e construir a mensagem
- Enviar a mensagem ao servidor.

c) **(20%) Cliente: leitura de uma mensagem do servidor.** O cliente tem que fazer duas coisas aparentemente em simultâneo: ler o teclado para obter o texto a enviar ao servidor, e ao mesmo tempo, estar à "escuta" de mensagens vindas do servidor. A situação é semelhante à existente no trabalho prático. **Sugestão:** o servidor avisa o cliente através de sinais antes de lhe enviar uma mensagem, permitindo ao cliente "concentrar-se" na leitura do teclado.

Escreva a parte do **cliente** responsável por:

- Atender/ler uma mensagem enviada pelo servidor (o texto é impresso no ecrã).
- Se usar sinais, então tem que incluir a preparação/configuração do cliente quanto à recepção desses sinais.
- Abertura/preparação do *pipe* do servidor.

O código de cada um destes pontos pode estar em sítios diferentes. Deve identificar bem o local de cada código.
A criação do named pipe do cliente não faz parte desta alínea.

d) **(20%) Servidor**: Atendimento de um cliente. O servidor deve manter uma estrutura de dados (tabela?) para os clientes conhecidos. Cada mensagem recebida terá como efeito percorrer essa estrutura de dados e enviar o texto acabado de receber a cada um dos outros clientes conhecidos. Cada mensagem recebida tanto pode ser de um novo cliente, ou de um já conhecido. Deve analisar (e actualizar) a estrutura de dados dos clientes para gerir este aspecto.

Escreva a parte do **servidor** responsável por:

- Definição e inicialização da estrutura onde guarda informação acerca dos clientes conhecidos. Pode assumir um máximo de 30 clientes.
- Recepção de uma mensagem e replicação dessa mensagem aos clientes conhecidos.
- Manutenção/actualização da estrutura de dados acerca dos clientes conhecidos.

Sugestão: neste caso é mais simples abrir e fechar o *named pipe* do cliente a cada mensagem a enviar (em vez de manter o *pipe* dos clientes permanentemente abertos). Vai ser dada *alguma* tolerância quanto a possíveis situações de espera mútua (*deadlock*) nos *pipes*, desde que não se cometam erros demasiado crassos.

e) **(20%) Servidor: verificação dos clientes que ainda existem/já saíram.** O servidor vai periodicamente (de 30 em 30 segundos) averiguar quais os clientes que enviaram mensagens. Aqueles que não enviaram serão "esquecidos" (a sua informação é removida da estrutura de dados que descreve os clientes conhecidos).

Sugestão. Nesta questão, a parte de algoritmo não é o tema central e por isso fornece-se:

> Na estrutura de dados acerca dos clientes, irá existir uma *flag* "enviou mensagem". Sempre que um cliente envia uma mensagem, a *flag* relativa a ele a é posta a 1. De 30 em 30 segundos, o servidor vai averiguar o estado da *flag* para cada cliente. Para aqueles em que a *flag* esteja a 0 (não foi recebida qualquer mensagem), vai remover os dados desse cliente. As *flags* são postas todas a 0 e o procedimento repete-se dai a 30 segundos.

Escreva a parte do **servidor** responsável por:

- Colocar a flag rlatica a um cliente a 1 quando recebe uma mensagem desse cliente (uma linha de código: qual e onde fica). Relativa
- Analisar as *flags* de 30 em 30 segundos e actualizar a estrutura de dados acerca dos clientes conhecidos.
- Garantir que essa análise é executada automaticamente a cada 30 segundos mas sem, no entanto, impedir que o servidor esteja a trabalhar e receber/reenviar mensagens.

Dica: O trabalho prático teve um mecanismo de temporização semelhante – é usar a mesma estratégia.