Laboratórios de Sistemas Operativos

Tutorial #7: Comunicação entre processos usando *pipes*

Neste tutorial vamos explorar a programação de comunicação entre processos usando *pipes*. Começaremos pelo uso de *pipes* simples para comunicação entre um processo pai e um processo filho, depois suportaremos comunicação entre quaisquer processos usando *named pipes*.

1. Comunicação entre processos pai e filho usando *pipes* simples

1. Estude o programa pipes.c, no qual um processo pai envia uma sequência de mensagens a um processo filho, que por sua vez as imprime no *stdout.*
2. Compile o programa e experimente executá-lo, confirmando que o *output* é o esperado.
3. Experimentarem sleep entre mensagens escritas, confirmar que fronteiras entre mensagens se perdem.
4. Resposta do filho: a cada 10 bytes lidos, “ack”. Experimentar usando o mesmo pipe (colocando sleep no pai), observar que corre mal, porquê? Resolver usando 2 pipes.

2. Comunicação entre processos usando *named pipes.*

1. Estude o programa named_pipes.c, que implementa a mesma lógica da 1ª alínea da secção anterior, mas agora entre dois processos lançados autonomamente. Para tal, recorre a *named pipes*.
2. Compile o programa e experimente executá-lo (lançando ambos os processos), confirmando que o *output* é o esperado.
3. Tal como antes, pretende-se que, a cada 10 bytes recebidos pelo processo que corre o programa *receiver*, este devolva uma resposta ao processo que corre o programa *sender*. Tal como com *pipes* simple, os *named pipes* são unidirecionais, logo será preciso recorrer a dois *named pipes* para suportar este novo comportamento.
4. Implemente esta variante usando um segundo *named pipe* que é criado pelo processo emissor com o nome “/tmp/acks-pipe”. Assuma que o código de ambos os programas conhecem o nome desse *pipe* (declare-o simplesmente numa macro/constante).
5. Pretende-se agora uma variante mais flexível em que o nome do *pipe* para as respostas “ack” é passado como argumento de linha de comandos do processo emissor, sendo o processo emissor que cria o *named pipe*. Neste caso, o nome do *named pipe* é previamente desconhecido do processo *receiver*, logo o processo *sender* deve enviar o nome do novo *pipe* ao processo *receiver* para que este o possa abrir.