Roteiro de Estudos 9
Pipes

1. **LEITURA**
   - **Celso** A. S. Santos, **Programação em tempo real**,
     - Capítulo 4 (Pipes)

2. **RESUMÃO sobre Pipes**
   - Slides com uma compilação do conteúdo e **exemplos adicionais**

3. **VÍDEOS**
   - Communicating between processes (using pipes) in C (3’59”)
   - Simulating the pipe "|" operator in C (19’51”)
   - Working with multiple pipes (19’37”)
   - Practical use case for fork and pipe in C (12’51”)
   - Two way communication between processes (using pipes) in C (18’15”)
   - How to send a string through a pipe (12’08”)
   - How to send an array through a pipe (15’18”)

4. **EXERCÍCIOS (valendo turings!!)**
   - **Responda o formulário fornecido juntamente com este roteiro**

==========

Lista de Exercícios de Consolidação

O objetivo da lista é ajudar no estudo individual dos alunos. Soluções de questões específicas poderão ser discutidas em sala de aula, conforme interesse dos alunos.

==========

1. Explicite as diferenças entre pipes e fifos (também denominados *named pipes*) em Linux. Descreva, para cada um dos tipos de comunicação, em qual situação cada uma pode ser utilizada.

2. Sobre o código a seguir, responda:

```
#define SIZE 6
#define READ 0
#define WRITE 1
main()
{
       pid_t pid1, pid2; int status;
       int fd[2]; char buffer[SIZE+1];
       struct rusage usage;
       pipe(fd);
              if ((pid1=fork())==0) { // child 1
                     while(1) {
                     read(fd[READ], buffer, SIZE);
                     buffer[SIZE]='\0';
                     write(fd[WRITE], "tomato", SIZE);
              }
       } else if ((pid2=fork())==0) { // child 2
              while(1) {
                     read(fd[READ], buffer, 6);
                     buffer[6]='\0';
                     write(fd[WRITE], "turnip", 6);
              }
       } else { // parent
              write(fd[WRITE],"potato",6);
              fprintf(stderr,"Parent: I wrote a potato!\n");
              sleep(3);
              read(fd[READ], buffer, 6); buffer[6]='\0';
              fprintf(stderr,"Parent: I got back a %s!\n", buffer);
              kill(pid1, SIGINT); waitpid(pid1, &status, NULL);
              kill(pid2, SIGINT); waitpid(pid2, &status, NULL);
       }
}
```

a) O que este código faz?

b) O que acontece se omitirmos o comando write do pai?

c) É possível prever qual será a saída do programa?

d) O que acontece se omitirmos o comando write do um dos filhos?

3. Descreva sucintamente os principais passos do código que a shell executa para efetuar as operações do seguinte comando encadeado:

```
ls -l | sort | grep batata
```

4. Como você faria para implementar o seguinte programa:

   Crie 1 processo *master* e 10 processos *racer* em que o processo *master* está alimentando um pipe com "Win", e então transmite o sinal SIGUSR1 para seu grupo de processos. Quando um processo *racer* recebe o sinal SIGUSR1, ele tenta ler o pipe, aquele que conseguir ler a mensagem "Win" ganha aquela rodada e para de disputar a corrida. No final do jogo, terão acontecido 10 rodadas, e cada processo possuirá um número que é o número da rodada em que ele ganhou a corrida (i.e. pegou a mensagem "Win").