Comunicação entre Processos

Última aula...

Usamos a chamada de sistema **fork**( ) para criar processos filhos

Entendemos como é o controle de execução dos processos pai e filho na programação

- Além disso, através da chamada de sistema fork( ) podemos dividir um determinado trabalho entre 2 processos e implementar um programa paralelo para MELHORAR o desempenho. Exemplo: criar um código para:
  a) Somar os elementos do vetor
  b) Encontrar o menor elemento do vetor

0	1	2	3	...	N-1
10	2	13	44	...	98

Qual seria a estratégia?

0	1	2	3	...	N-1
10	2	13	44	...	98

Atividade

- Usando o exemplo fork( ) da última aula, defina sua estratégia e faça seu programa para testar o processamento paralelo entre os processos

- Para declarar o vetor em C use:

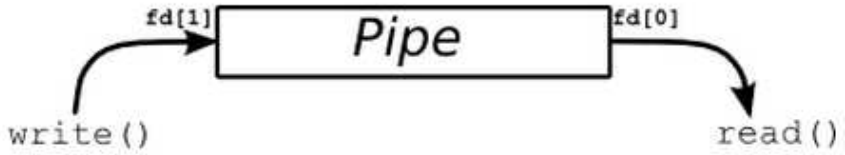
```
int vet[10] = {1,4,5,3,22,7,9,0,11,44}
```

Pergunta-se

- Como os processos, pai e filho, podem se comunicar para que um deles possa enviar uma mensagem para o outro?

- **Exemplo**: ps aux | grep chrome

Pipe



- Pipe em inglês quer dizer tubo, cano... e conectam um processo a outro, diretamente.
- Esse pipe tem um sentido (half-duplex). Entra informação de um lado e sai do outro.
- O pipe é criado com um vetor de duas posições:
  - **Posição 0**: leitura
  - **Posição 1**: escrita
- Precisaremos usar as chamadas **write()** e **read()** para enviar e receber informações pelo pipe!

```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
int main(void){
        int     fd[2], nbytes;
        pid_t   childpid;
        char    string[] = "Hello, world!\n";
        char    readbuffer[80];
        pipe(fd);
        if((childpid = fork()) == -1){
                perror("fork");
                exit(1);
        }
        if(childpid == 0){
                /* Child process closes up input side of pipe */
                close(fd[0]);
                /* Send "string" through the output side of pipe */
                write(fd[1], string, (strlen(string)+1));
                exit(0);
        }
        else{
                /* Parent process closes up output side of pipe */
                close(fd[1]);
                /* Read in a string from the pipe */
                nbytes = read(fd[0], readbuffer, sizeof(readbuffer));
                printf("Received string: %s", readbuffer);
        }
        return(0);
}
```

Exemplo

(comunicação entre processos)

Atividade

Utilizando o exemplo com o fork() e mais o pipe(), implemente um código para efetuar a soma do vetor definido no slide 4.