Disciplina: Sistema Operacionais Embarcados **Código:** 120961 **Turma:** A
Professor: Diogo Caetano Garcia
Aluno/Matrícula: Fábio Barbosa Pinto – 11/0116356

Questionário: 07_Pipes_Sinais_Alarmes_2

1. Crie um programa em C que cria um processo-filho e um pipe de comunicação. Faça com que o processo-pai envie os valores 1, 2, 3, 4, 5, 6, 7, 8, 9 e 10 para o processo-filho, com intervalos de 1 segundo entre cada envio. Depois de o processo-pai enviar o número 10, ele aguarda 1 segundo e termina a execução. O processo-filho escreve na tela cada valor recebido, e quando ele receber o valor 10, ele termina a execução.

```
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <unistd.h>
#include <string.h>

#define PARADA 10

int main()
{
	int pid, inteiro;
	int fd[2];
	//cria o pipe
	pipe(fd);
	//duplica os processos
	pid = fork();
	if (pid == 0)//processo filho
	{
		for (int i = 0; i < PARADA; ++i)
		{
			if(read(fd[0], &inteiro, sizeof(inteiro)) < 0)
				printf("Erro na leitura do pipe\n");
			else
				printf("%d\n", inteiro);
		}
	}
	else//processo pai
	{
		int i;
		for(i = 0; i < PARADA; i++)
		{
			inteiro = (i+1);
			if (write(fd[1], &inteiro, sizeof(inteiro)) < 0)
```

```
                {
                        printf("Erro na escrita do pipe\n");

                }
                sleep(1);
        }

    }

    return 0;

}
```

2. Crie um programa em C que cria um processo-filho e um pipe de comunicação. Utilize o pipe para executar a seguinte conversa entre processos:

FILHO: Pai, qual é a verdadeira essência da sabedoria?
PAI: Não façais nada violento, praticai somente aquilo que é justo e equilibrado.
FILHO: Mas até uma criança de três anos sabe disso!
PAI: Sim, mas é uma coisa difícil de ser praticada até mesmo por um velho como eu...

```
#include <stdio.h>
#include <signal.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include  <sys/types.h>

int main(){
char msg1p[100];
char msg2p[100];
char msg1f[100];
char msg2f[100];

int i;
int pid;
int fd[2];
pipe(fd);
pid=fork();
        if(pid == 0){
sprintf(msg1f, "FILHO: Pai, qual é a verdadeira essência da sabedoria? \n");
write(fd[1], msg1f, strlen(msg1f)+1);
usleep(500000);}
else{
i = -1;
do{
```

```
i++;
read(fd[0], msg1f+i, 1);}
while(msg1f[i] != '\0');
printf("\n%s", msg1f);
sprintf(msg1p, "PAI: Não façais nada violento, praticai somente aquilo que é justo e equilibrado.");
write(fd[1], msg1p, strlen(msg1p)+1);
usleep(1000000);
}
        if(pid == 0){
i = -1;
do{
i++;
read(fd[0], msg1p+i, 1);}
while(msg1p[i] != '\0');
printf("\n%s", msg1p);
sprintf(msg2f, "FILHO: Mas até uma criança de três anos sabe disso! \n");
write(fd[1], msg2f, strlen(msg2f)+1);
usleep(1500000);}
else{
usleep(2000000);
i = -1;
do{
i++;
read(fd[0], msg2f+i, 1);}
while(msg2f[i] != '\0');
printf("\n%s", msg2f);
puts( "PAI: Sim, mas é uma coisa difícil de ser praticada até mesmo por um velho como eu...");
}
wait(NULL);
return 0;
}
```

3. Crie um programa em C que cria dois processos-filhos e um pipe de comunicação. Utilize o pipe para executar a seguinte conversa entre processos:

```
char msgs[500];
int i;
int pid[2];
int fd[2];
pipe(fd);
pid[0] = fork();
if (pid[0]==0)
{
sprintf(msgs, "FILHO1: Quando o vento passa, é a bandeira que se move.");
write(fd[1], msgs, strlen(msgs)+1);
usleep(500000);
```

```
}
else
{
pid[1]=fork();
if(pid[1]==0)
{
i=-1;
do
{
i++;
read(fd[0], msgs+i, 1);
}while(msgs[i]!='\0');
printf(" %s\n", msgs);
sprintf(msgs, "FILHO2: Não, é o vento que se move.");//
write(fd[1], msgs, strlen(msgs)+1);
}
else
{
usleep(500000);
}
//pai:
i=-1;
do
{
i++;
read(fd[0], msgs+i, 1);
}while(msgs[i]!='\0');
printf(" %s\n", msgs);
puts("PAI: Os dois se enganam. É a mente dos senhores que se move.");//escrever direto
}
exit(0);
}
```

4. Crie um programa em C que cria um processo-filho e um pipe de comunicação. O processo-filho deverá pedir o nome do usuário, envia-lo para o pai via pipe, e o pai deverá escrever o nome do usuário no terminal.

```
#include <stdio.h>
#include <signal.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include  <sys/types.h>

int main(){
char nomef[201];
```

```
char usuario[200];
int i;
int pid;
int fd[2];
pipe(fd);
pid = fork();
        if (pid ==0){
        printf("\nDigite seu nome:\n");
        gets(usuario);
        sprintf(nomef, usuario);
        write(fd[1], nomef, strlen(nomef)+1);
        usleep(500000);}
else{
i = -1;
do{
i++;
read(fd[0], nomef+i, 1);
}while(nomef[i]!='\0');
printf("Nome pelo pai: %s\n", nomef);
}
return 0;}
```

5. Utilize o sinal de alarme para chamar a cada segundo o comando ps ordenando todos os processos de acordo com o uso da CPU. Ou seja, seu código atualizará a lista de processos a cada segundo. Além disso, o código deverá tratar o sinal do CTRL-C, escrevendo "Processo terminado!" na tela antes de terminar a execução do processo.

```
        #include <stdio.h>
        #include <stdlib.h>
        #include <unistd.h>

        #include <sys/types.h>
        #include <sys/stat.h>

        void tratamento_alarme(int sig)
        {
                alarm(1);
                system("ps --sort=-pcpu");
        }

        void acaba_prog(int sig)
        {
                printf("Processo Terminado\n");
                exit(0);
        }

        int main()
```

```
{
        signal(SIGALRM, tratamento_alarme);
        signal(SIGINT, acaba_prog);
        alarm(1);
        printf("Aperte CTRL+C para acabar:\n");
        while(1);
        return 0;
}
```