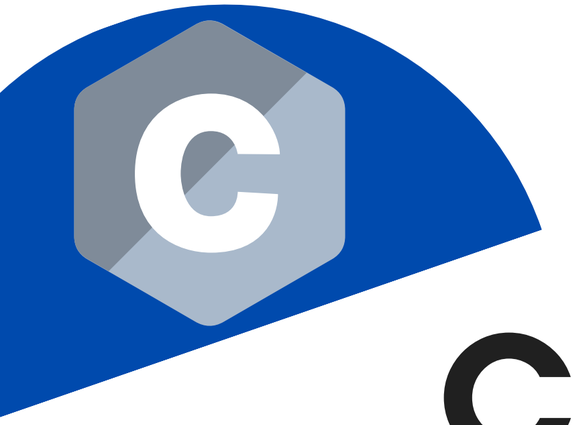

Comunicação entre Processos

(usando a API do kernel e a Standard C Library - Parte II)

Bem-vindos à aula!

Agenda de Hoje

- Uso de sinais
- Handlers
- Variável errno
- Funções perror() e strerror()

O que são sinais?

- **Definição:** Sinais são interrupções que podem ser enviadas a um processo para notificar sobre eventos assíncronos. Cada sinal tem um nome simbólico, geralmente começando com o prefixo **SIG**. Vêm listados no ficheiro de cabeçalho **<signal.h>**.

- **Origem:** Sinais podem ser gerados por várias fontes, como o teclado (ex.: Ctrl+C), hardware (ex.: divisão por zero), serviços do sistema operativo (ex.: kill(), alarm()), ou comandos da shell.

Utilidade dos sinais no sistema operativo

- **Interrupção de processos:** Sinais podem interromper a execução de um processo para tratar eventos como erros ou interrupções do usuário.
- **Sincronização:** Eles são usados para sincronizar processos, permitindo que um processo notifique outro sobre a conclusão de uma tarefa ou a ocorrência de um evento.
- **Gestão de processos:** Sinais como SIGKILL e SIGTERM são usados para terminar processos de forma controlada ou forçada

Sinais Reprogramáveis e Não Reprogramáveis

- **Reprogramáveis:** A maioria dos sinais pode ser capturada e tratada por handlers personalizados. Exemplos incluem SIGHUP, SIGINT, SIGUSR1, SIGUSR2, entre outros.

- **Não Reprogramáveis:** Alguns sinais não podem ser capturados ou ignorados, como SIGKILL e SIGSTOP. Estes sinais são usados pelo sistema operativo para garantir que processos possam ser terminados ou suspensos de forma confiável.

Sinais e seus Handlers

- **Handlers de Sinais:** Um handler de sinal é uma função que é executada quando um processo recebe um sinal específico. Você pode definir handlers personalizados para tratar sinais de maneira específica.

- **Definição de Handlers:** Em C, você pode usar a função **signal()** para definir um handler para um sinal. Por exemplo:

```
#include <signal.h>

void meu_handler(int signo) {
   // Código para tratar o sinal
}

signal(SIGINT, meu_handler); // Define meu_handler para tratar
```

Exemplos de sinais comuns

Nome	Descrição	Origem
`SIGHUP`	Recarregar ou reiniciar o processo sem interromper.	Fechamento de terminal, logout
`SIGTSTP`	Pausa temporária de execução.	Teclado (^z)
`SIGCHLD`	Notifica o pai quando um processo filho termina ou muda de estado.	SO
`SIGINT`	Interrupção de um processo em execução.	Teclado (^c)
`SIGKILL`	Finalização imediata de um processo.	SO
`SIGUSR1 e SIGUSR2`	Sinais personalizados para comunicação entre processos.	Processo

A Variável errno

O que é?

- **errno** é uma variável global definida no cabeçalho <errno.h>.
- Após a execução de uma system call ou da biblioteca padrão que falhou, **errno** é configurada com um código de erro específico.
- Cada valor de **errno** representa um tipo de erro padrão (ex.: EACCES para "Permissão negada", ENOENT para "Arquivo não encontrado").

```
#include <stdio.h>
#include <errno.h>

int main() {
    FILE *file = fopen("inexistente.txt", "r");
    if (file == NULL) {
        printf("Erro ao abrir ficheiro. errno: %d\n", errno);
    }
    return 0;
}
```

Função perror()

O que é?

- **perror()** é usada para imprimir uma mensagem de erro legível, com base no valor atual de **errno**.
- Ela combina uma mensagem customizada fornecida pelo programador com uma descrição padrão do erro correspondente ao valor de **errno**.

```
#include <stdio.h>
#include <errno.h>

int main() {
    FILE *file = fopen("inexistente.txt", "r");
    if (file == NULL) {
        perror("Erro ao abrir arquivo");
    }
    return 0;
}
```

Função strerror()

O que é?

- **strerror()** retorna uma string com a descrição do erro correspondente ao valor de **errno**.
- Diferentemente de **perror()**, ela apenas retorna a mensagem de erro, sem imprimir diretamente.

```
#include <stdio.h>
#include <errno.h>
#include <string.h>

int main() {
    FILE *file = fopen("inexistente.txt", "r");
    if (file == NULL) {
        printf("Erro ao abrir arquivo: %s\n", strerror(errno));
    }
    return 0;
}
```

Tente e Aprenda

Hora da Atividade

Ficha 6 - Exercícios 3, 4, 5, 6

E terminamos!