Recursos do Sistema Linux

Sistemas Embarcados

Conteúdo

1. Acesso a arquivos
2. Processos e Sinais
3. Threads POSIX e biblioteca *pthreads*
4. Comunicação e Sincronismo entre Processos
5. Programação usando *sockets*
6. Device Drivers

Acesso a Arquivos

1. **Arquivo regular**: tipo comum que contém dados somente. Os arquivos regulares podem ser dos mais variados tipos, guardando os mais diferentes tipos de informações. Existem arquivos de áudio, vídeo, imagem, texto, enfim. Os arquivos se dividem em 2 (duas) categorias principais:
   - **Binários**: arquivos binários são compostos por bits 1 e 0 e só podem ser interpretados pelo sistema operacional. Programas e bibliotecas são exemplos de arquivos binários;
   - **Texto**: arquivos do tipo texto são compostos por informações em forma de texto, que podem ser entendidas pelo usuário comum. Arquivos desse tipo não necessariamente contém texto propriamente dito. Quando se diz que um arquivo é do tipo texto isto significa que, se aberto em um editor de texto, serão exibidos informações legíveis (ainda que possam não fazer muito sentido).

Acesso a Arquivos

2. **Diretórios**: os diretórios são utilizados para separar um grupo de arquivos de outros. Um diretório pode conter arquivos e outros diretórios, que serão chamados subdiretórios;
3. **Dispositivos**: todo componente de hardware instalável é chamado dispositivo. Placas de vídeo, som, rede, drives de CD-ROM, tudo o que se liga na interface USB do PC, memória RAM, são dispositivos. Os dispositivos podem ser:
   - **De bloco**: dispositivos de bloco utilizam buffer para leitura/gravação. Geralmente são unidades de disco, como HD's, CD's, etc;
   - **De caracter**: esses dispositivos não utilizam buffer para leitura/ gravação. A maioria dos dispositivos PCI e outros dispositivos como impressoras, mouse, etc. são do tipo caracter;
   - **Fifo**: trata-se de um canal de comunicação, através do qual pode-se ver os dados que estão trafegando por um dispositivo;

Acesso a Arquivos

4. **Links**: os links são arquivos utilizados para fazer referência a um outro arquivo localizado em outro local. Em outras palavras, são atalhos. Os links podem ser de 2 (dois) tipos:
   - **Simbólicos**: fazem uma referência ao arquivo através de seu endereço lógico no disco ou memória. São os links mais comuns;
   - **Absolutos**: fazem referência ao arquivo através do seu endereço físico no disco rígido ou memória.
5. **FIFO**: canal de comunicação, utilizado para direcionar os dados produzidos por um processos para um outro processo.

stdio.h

Declaração de Arquivos:

```
FILE *ponteiro_para_arquivo;
```

Função para abrir arquivos

```
FILE *fopen (char *nome_do_arquivo, char
*modo)
```

Função para Fechar arquivos

```
int fclose (FILE *ponteiro_para_arquivo);
```

Buffer de gravação (in/out)

stdio.h

Modo	*Significado*
"r"	Abre um arquivo texto para leitura. O arquivo deve existir antes de ser aberto.
"w"	Abrir um arquivo texto para gravação. Se o arquivo não existir, ele será criado. Se já existir, o conteúdo anterior será destruído.
"a"	Abrir um arquivo texto para gravação. Os dados serão adicionados no fim do arquivo ("append"), se ele já existir, ou um novo arquivo será criado, no caso de arquivo não existente anteriormente.
"rb"	Abre um arquivo binário para leitura. Igual ao modo "r" anterior, só que o arquivo é binário.
"wb"	Cria um arquivo binário para escrita, como no modo "w" anterior, só que o arquivo é binário.
"ab"	Acrescenta dados binários no fim do arquivo, como no modo "a" anterior, só que o arquivo é binário.

stdio.h

Modo	*Significado*
"r+"	Abre um arquivo texto para leitura e gravação. O arquivo deve existir e pode ser modificado.
"w+"	Cria um arquivo texto para leitura e gravação. Se o arquivo existir, o conteúdo anterior será destruído. Se não existir, será criado.
"a+"	Abre um arquivo texto para gravação e leitura. Os dados serão adicionados no fim do arquivo se ele já existir, ou um novo arquivo será criado, no caso de arquivo não existente anteriormente.
"r+b"	Abre um arquivo binário para leitura e escrita. O mesmo que "r+" acima, só que o arquivo é binário.
"w+b"	Cria um arquivo binário para leitura e escrita. O mesmo que "w+" acima, só que o arquivo é binário.
"a+b"	Acrescenta dados ou cria uma arquivo binário para leitura e escrita. O mesmo que "a+" acima, só que o arquivo é binário

stdio.h

```c
#include <stdio.h>

int main(int argc, const char * argv[]) {

    FILE *fp; // Declaração da estrutura
    fp=fopen ("exemplo.bin","wb");  /* o arquivo se chama
                                       exemplo.bin e está
                                       localizado no
                                       diretório corrente */
    if (!fp)
        printf ("Erro na abertura do arquivo.");

    return 0;
}
```

!fp = ponteiro para o arquivo

A posição atual na leitura do arquivo é armazenada a cada leitura.

fseek()

stdlib.h

Abortar a execução de um programa:

```
void exit (int codigo_de_retorno);
```

```c
#include <stdio.h>
#include <stdlib.h> /* Para a função exit() */

int main(int argc, const char * argv[]) {

    FILE *fp;
    fp=fopen ("exemplo.bin","wb");
    if (!fp)
    {
        printf ("Erro na abertura do arquivo. Fim de programa.");
        exit (1);
    }
    return 0;
}
```

stdio.h

Escrita em arquivos:

```
int putc (int ch, FILE *fp);
```

```c
#include <stdio.h>
#include <stdlib.h>
int main(int argc, const char * argv[]) {

    FILE *fp;
    char string[100];
    int i;
    fp = fopen("arquivo.txt","w");
    if(!fp)
    {
        /* Arquivo ASCII, para escrita */
        printf( "Erro na abertura do arquivo");
        exit(0); }
    printf("Entre com a string a ser gravada no arquivo:");
    gets(string);
    for(i=0; string[i]; i++) putc(string[i], fp); /* Grava a string, caractere a
                                                     caractere */
    fclose(fp);
    return 0;
}
```

stdio.h

Leitura de arquivos:

```
int getc (FILE *fp);
```

```c
#include <stdio.h>
#include <stdlib.h>
int main(int argc, const char * argv[]) {

    FILE *fp;
    char c;
    fp = fopen("arquivo.txt","r"); /* Arquivo ASCII, para leitura */
    if(!fp)
    {
        printf( "Erro na abertura do arquivo");
        exit(-1);
    }
    while((c = getc(fp) ) != EOF)  /* Enquanto não chegar ao final do arquivo */
        printf("%c", c);           /* imprime o caracter lido */
    fclose(fp);
    return 0;
}
```

stdio.h

Detectar fim de arquivos:

```
int feof (FILE *fp);
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(int argc, const char * argv[]) {
    FILE *p;
    char c, str[30], frase[80] = "Este e um arquivo chamado: ";
    int i;
    /* Le um nome para o arquivo a ser aberto: */
    printf("\n\n Entre com um nome para o arquivo:\n");
    gets(str);
    if (!(p = fopen(str,"w")))  /* Caso ocorra algum erro na abertura do arquivo..*/
    {                           /* o programa aborta automaticamente */
        printf("Erro! Impossivel abrir o arquivo!\n");
        exit(1); }
    /* Se nao houve erro, imprime no arquivo e o fecha ...*/
    strcat(frase, str);
    for (i=0; frase[i]; i++)
        putc(frase[i],p);
    fclose(p);
    /* Abre novamente para  leitura  */
    p = fopen(str,"r");
    c = getc(p);            /* Le o primeiro caracter */
    while (!feof(p)) {       /* Enquanto não se chegar no final do arquivo */
        printf("%c",c);     /*   Imprime o caracter na tela */
        c = getc(p);        /* Le um novo caracter no arquivo */
    }
    fclose(p);              /* Fecha o arquivo */
}
```

Arquivos Padrão

* **stdin**: dispositivo de entrada padrão (geralmente o teclado)

* **stdout**: dispositivo de saída padrão (geralmente o vídeo)

* **stderr**: dispositivo de saída de erro padrão (geralmente o vídeo)

* **stdaux**: dispositivo de saída auxiliar (em muitos sistemas, associado à porta serial)

* **stdprn** : dispositivo de impressão padrão (em muitos sistemas, associado à porta paralela)

Arquivos Padrão

Exemplo de uso:

```
ch =getc(stdin);
```

```
putc(ch, stdout);
```

Mais funções

Lê uma string de um *stream*

```
char *fgets (char *str, int tamanho,FILE *fp);
```

Escreve um string em um *stream*

```
char *fputs (char *str,FILE *fp);
```

Verifica o indicador de erro

```
int ferror (FILE *fp);
```

Imprime a mensagem de erro

```
void perror ( const char * str );
```

Mais funções

```c
#include <stdio.h>
#include <stdlib.h>
int main()
{
    FILE *pf;
    char string[100];
    if((pf = fopen("arquivo.txt","w")) ==NULL)
    {
        printf("\nNao consigo abrir o arquivo ! ");
        exit(1); }
    do {
        printf("\nDigite uma nova string. Para terminar, digite <enter>: ");
        gets(string);
        fputs(string, pf);
        putc('\n', pf);
        if(ferror(pf))
        {
            perror("Erro na gravacao");
            fclose(pf);
            exit(1);
        }
    } while (strlen(string) > 0);
    fclose(pf);
}
```

Mais funções

Lê bloco de dados do *stream*

```
unsigned fread (void *buffer, int numero_de_bytes, int
count, FILE *fp);
```

Escreve bloco de dados em um *stream*

```
unsigned fwrite (void *buffer,int numero_de_bytes,int
count,FILE *fp);
```

Mais funções

```c
#include <stdio.h>
#include <stdlib.h>
int main()
{
    FILE *pf;
    float pi = 3.1415;
    float pilido;
    if((pf = fopen("arquivo.bin", "wb")) == NULL) /* Abre arquivo binário para escrita */
    {
        printf("Erro na abertura do arquivo");
        exit(1);
    }
    if(fwrite(&pi, sizeof(float), 1,pf) != 1)
        printf("Erro na escrita do arquivo");
    fclose(pf);

    if((pf = fopen("arquivo.bin", "rb")) == NULL) /* Abre o arquivo novamente para leitura */
    {
        printf("Erro na abertura do arquivo");
        exit(1);
    }
    if(fread(&pilido, sizeof(float), 1,pf) != 1) /* Le em pilido o valor da variável
armazenada anteriormente */
        printf("Erro na leitura do arquivo");
    printf("\nO valor de PI, lido do arquivo e': %f", pilido);
    fclose(pf);
    return(0);
}
```

Mais funções

Reposiciona o ponteiro do *stream*

```
int fseek ( FILE * stream, long int offset, int origin );
```

Nome	*Valor*	*Significado*
SEEK_SET	0	Início do arquivo
SEEK_CUR	1	Posição corrente do arquivo
SEEK_END	2	Fim do arquivo

Volta ponteiro do *stream* para o início

```
void rewind ( FILE * stream );
```

Apara um arquivo

```
int remove (char *nome_do_arquivo);
```

Mais funções

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
int main()
{
    FILE *p;
    char str[30], frase[] = "Este e um arquivo chamado: ", resposta[80];
    int i;
    /* Le um nome para o arquivo a ser aberto: */
    printf("\n\n Entre com um nome para o arquivo:\n");
    fgets(str,29,stdin);/* Usa fgets como se fosse gets */
    for(i=0; str[i]; i++)
        if(str[i]=='\n')
            str[i]=0; /* Elimina o \n da string lida */
    if (!(p = fopen(str,"w"))) /* Caso ocorra algum erro na abertura do arquivo..*/
    {
        printf("Erro! Impossivel abrir o arquivo!\n");
        exit(1);
    }
    fputs(frase, p);
    fputs(str,p);
    fclose(p);
    /* abre novamente e le */
    p = fopen(str,"r");
    fgets(resposta, 79, p);
    printf("\n\n%s\n", resposta);
    fclose(p);
    remove(str);
    return(0);
}
```

Mais funções

Escreve caracteres formatados em *stdout*

```
int printf (char *str,...);
```

Escreve caracteres formatados em uma *string*

```
int sprintf ( char * str, const char * format, ... );
```

Escreve caracteres formatados em *stream*

```
int fprintf ( FILE * stream, const char * format, ... );
```

Mais funções

Cód	*Formato*
%c	Um caracter (char)
%d	Um número inteiro decimal (int)
%i	O mesmo que %d
%e	Número em notação científica com o "e"minúsculo
%E	Número em notação científica com o "e"maiúsculo
%f	Ponto flutuante decimal
%g	Escolhe automaticamente o melhor entre %f e %e
%G	Escolhe automaticamente o melhor entre %f e %E

Cód	*Formato*
%o	Número octal
%s	String
%u	Decimal "unsigned" (sem sinal)
%x	Hexadecimal com letras minúsculas
%X	Hexadecimal com letras maiúsculas
%%	Imprime um %
%p	Ponteiro

Mais funções

Lê dados formatados de *stdin*

```
int scanf (char *str,...);
```

Lê dados formatados de uma *string*

```
int sscanf ( const char * s, const char * format, ...);
```

Lê dados formatados de *stream*

```
int fscanf ( FILE * stream, const char * format, ... );
```

Mais funções

```c
#include <stdio.h>
int main() {
    int i;
    char string1[20];
    printf( " Entre um valor inteiro: ");
    scanf("%d", &i);
    sprintf(string1,"Valor de i = %d", i);
    puts(string1);
    return 0;
}
```

Mais funções

```c
#include <stdio.h>
int main() {
    int i, j, k;
    char string1[]= "10 20 30";
    sscanf(string1, "%d %d %d", &i, &j, &k);
    printf("Valores lidos: %d, %d, %d", i, j, k);
    return 0;
}
```

Mais funções

```c
#include <stdio.h>
#include <stdlib.h>
int main() {
    FILE *p;
    char str[80],c;
    /* Le um nome para o arquivo a ser aberto: */
    printf("\n\n Entre com um nome para o arquivo:\n");
    gets(str);
    if (!(p = fopen(str,"w"))) /* Caso ocorra algum erro na abertura do
arquivo..*/
    {
        printf("Erro! Impossivel abrir o arquivo!\n");
        exit(1);
    }
    /* Se nao houve erro, imprime no arquivo, fecha ...*/
    fprintf(p,"Este e um arquivo chamado:\n%s\n", str);
    fclose(p);
    /* abre novamente para a leitura*/
    p = fopen(str,"r");
    while (!feof(p))
    {
        fscanf(p,"%c",&c);
        printf("%c",c);
    }
    fclose(p);
    return(0);
}
```