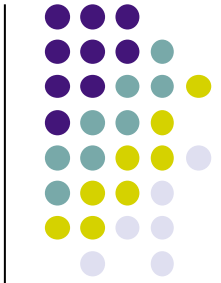
Sistemas de Computação

Sistemas de Arquivos

O Diretório de Trabalho



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

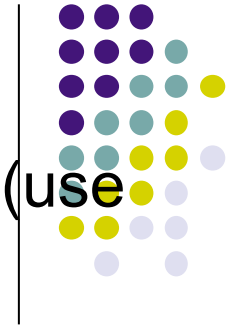
```
char *getwd(char *path)
```

- escreve em path o nome do diretório de traballho
- retorna o ponteiro para a string com o nome do diretório, ou NULL em caso de erro

```
int chdir(char *path)
```

- Troca para o diretório especificado em path
- Retorna 0 em caso de sucesso, e -1 em caso de falha

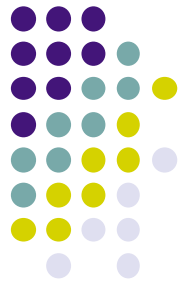
Percorrendo Diretórios



- Os arquivos de um diretório estão em um struct direct (use <sys/types.h> e <sys/dir.h>

```
 scandir(char *dirname, struct direct
**namelist, int (*select)(), int (*compar)())
```

- Retorna um array de ponteiros para entradas do diretório dirname (cada entrada está em `struct direct` ou `dirent`)
- namelist é um ponteiro para um vetor de ponteiros das entradas
- `select` é um ponteiro para uma função chamada para cada entrada (de diretório) e retorna não nulo se essa entrada deve ser incluida em namelist
- `compar` é um ponteiro para uma função para ordenar os elementos de namelist



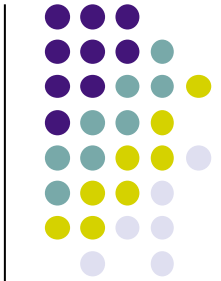
listaarquivos.c

```
#include <sys/types.h>
#include <sys/dir.h>
#include <sys/param.h>
#include <stdio.h>
extern  int alphasort();
char pathname[MAXPATHLEN];
main()   {
  int count,i;
  struct direct **files;
  int file_select();

  if (getwd(pathname) == NULL )  {  printf("Error getting path\n"); exit(0);
  }
  printf("Current Working Directory = %s\n",pathname);
  count = scandir( pathname, &files, file_select, alphasort);

  /* If no files found, make a non-selectable menu item */
  if (count <= 0) {    printf("No files in this directory\n"); exit(0);
  }
  printf("Number of files = %d\n",count);
  for (i=1;i<count+1;++i) printf("%s  ", files[i-1]->d_name);
  printf("\n"); /* flush buffer */
}
```

Exemplo de uma função de seleção

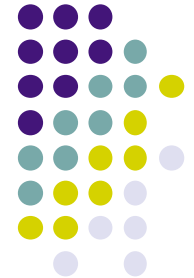


- scandir também retorna as entradas “.” e “..”
- Se quisermos suprimi-los precisamos declarar uma função de seleção

```
#define FALSE 0
#define TRUE 1

int file_select(struct direct   *entry)
{
    if ((strcmp(entry->d_name, ".") == 0) || (strcmp(entry->d_name, "..") == 0))
        return (FALSE);
    else
        return (TRUE);
}
```

Outra função de seleção

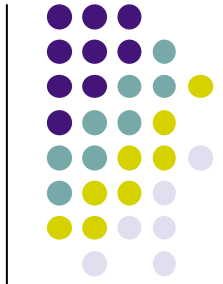


- Mostrar apenas os arquivos com sufixo .c .h e .o
- rindex(s,c) é uma função que retorna um pointeiro para a última ocorrência do caracter c em string s

```
int file_select ( struct direct   *entry)
{
   char *ptr;
   char *rindex(char *s, char c);

   if ((strcmp(entry->d_name, ".")== 0) || (strcmp(entry->d_name, "..") == 0))
        return (FALSE);
   /* Check for filename extensions */
   ptr = rindex(entry->d_name, '.')
   if ((ptr != NULL) && ((strcmp(ptr, ".c") == 0) || (strcmp(ptr, ".h") == 0) ||
      (strcmp(ptr, ".o") == 0) ))
              return (TRUE);
   else
              return(FALSE);
}
```

Outra forma de ler entradas de diretório



```
#include <sys/types.h>
#include <sys/dir.h> ou <dirent.h>
struct direct *readdir(char *dir)
```

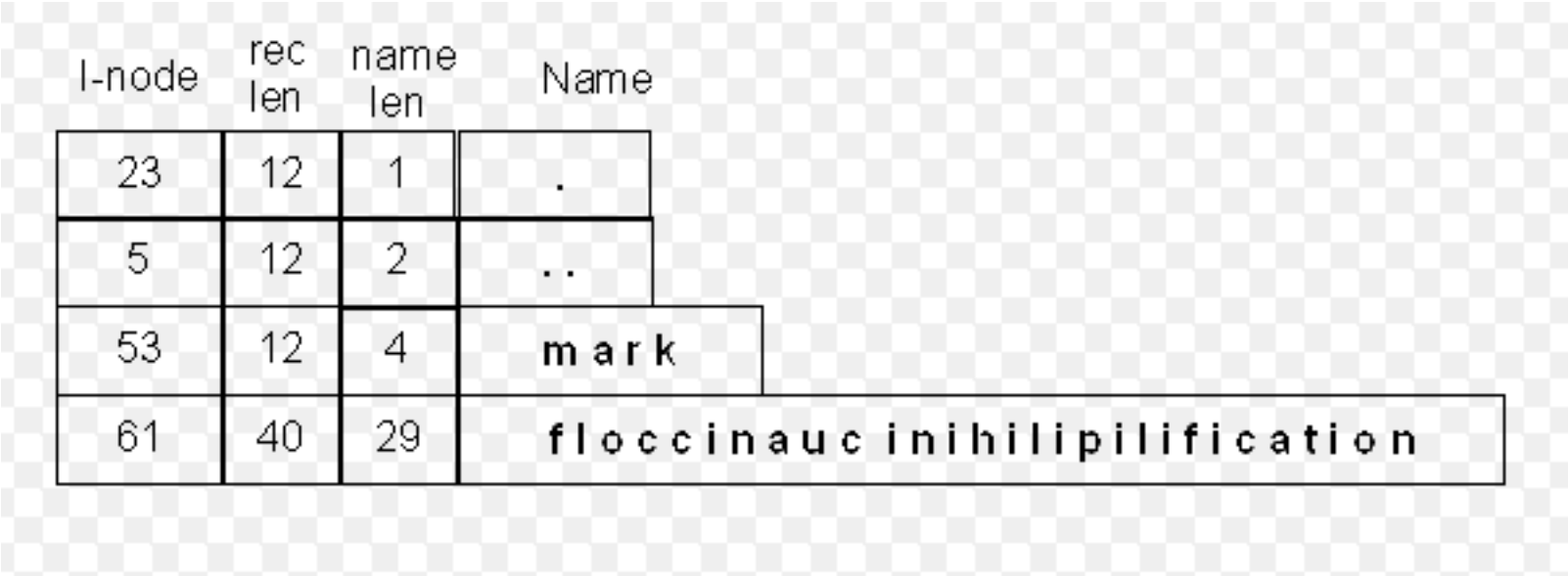
Ou

```
DIR *opendir(const char *name);
struct dirent *readddir(DIR *dirp)
```

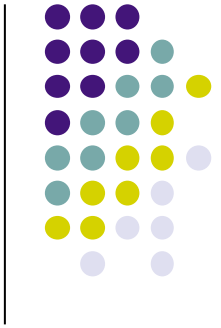
OBS: `DIR` é um directory stream e `dirent` é uma entrada no diretório

Informações na entrada de diretórios

```
struct dirent
{
    ino_t          d_ino;          /* inode number */
    off_t          d_off;          /* offset to the next dirent */
    unsigned short d_reclen;       /* length of this record */
    unsigned char  d_type;         /* type of file; not supported by all file systems */
    char           d_name[256];    /* filename */
};
```



Exemplo de readdir



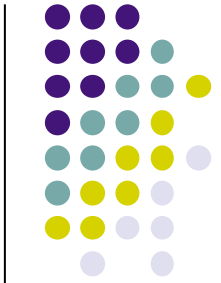
```
#include <sys/types.h>
#include <dirent.h>
#include <stdio.h>

int main(void)
{
   DIR *dir;
   struct dirent *entry;

   if (!(dir = opendir(".")))          return;
   if (!(entry = readdir(dir)))      return;

   do {
      if (entry->d_type == DT_DIR) printf("%s é diretorio\n",entry->d_name);
      else if (entry->d_type == DT_REG) printf("%s is file \n",entry->d_name);
      else if (entry->d_type == DT_SOCK) printf("%s is socket\n",entry->d_name);
      else if (entry->d_type == DT_LNK) printf ("%s is link\n",entry->d_name);
      else if (entry->d_type == DT_FIFO) printf ("%s is  named pipe\n",entry-
>d_name);
      else if (entry->d_type == DT_CHR) printf ("%s is char-device\n",entry->d_name);
      else if (entry->d_type == DT_BLK) printf ("%s is block-device\n",entry->d_name);
      else ("%s tem tipo default UNKNOWN\n",entry->d_name);
   } while (entry = readdir(dir));
}
```

Status de cada Arquivo



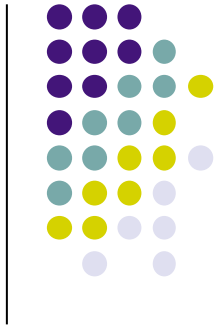
- Existem 2 funções úteis para se obter mais informações sobre cada arquivo: stat e fstat

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

int stat (char *pathcompleto, struct *buf)
int fstat (int fd, struct stat *buf)
```

- Retornam 0 em caso de sucesso, -1 em caso de falha
- buf é um ponteiro para a struct stat que contém informação sobre o arquivo

Informações na struct stat



```
struct stat {
        mode_t   st_mode;     /* File mode (type, perms) */
        ino_t    st_ino;      /* Inode number */
        dev_t    st_dev;      /* ID of device containing */
                              /* a directory entry for this file */
        dev_t    st_rdev;     /* ID of device */
                               /* This entry is defined only for */
                               /* char special or block special files */
        nlink_t  st_nlink;    /* Number of links */
        uid_t    st_uid;      /* User ID of the file's owner */
        gid_t    st_gid;      /* Group ID of the file's group */
        off_t    st_size;     /* File size in bytes */
        time_t   st_atime;    /* Time of last access */
        time_t   st_mtime;    /* Time of last data modification */
        time_t   st_ctime;    /* Time of last file status change */
                              /* Times measured in seconds since */
                               /* 00:00:00 UTC, Jan. 1, 1970 */
        long     st_blksize;  /* Preferred I/O block size */
        blkcnt_t st_blocks;   /* Number of 512 byte blocks allocated*/
}
```

Perguntas?

Exercícios!

1. Use o programa `listaarquivo.c` para imprimir o numero de inode, o tamanho (em bytes) e a idade em dias de cada arquivo do diretório corrente. Lembre-se que stat() espera o path completo do arquivo

```
Number of files = 7
SClient.java      inode 9636209   size: 2757 age: 1039 days
SServer.java      inode 9636210   size: 3699 age: 1039 days
listfiles         inode 44250424  size: 9492 age: 0 days
listfiles.c       inode 44248571  size: 2322 age: 0 days
signal.c          inode 10741963  size: 272 age: 945 days
tarfifador.c      inode 44215426  size: 783 age: 1 days
tarifador         inode 44215435  size: 9040 age: 1 days
```

2. Modifique o programa anterior para mostrar o número de links que cada arquivo possui. Depois, no diretório corrente, execute o o comando da shell

   ln <arquivo> <nomeLink> e veja o que mudou na saída de seu programa

Exercícios!

3. Escreva um programa que percorre recursivamente um diretório, a partir do diretório corrente, somando o tamanho de todos os arquivos encontrados.

4. Modifique o programa anterior para mostrar recursivamente os diretórios e arquivos aninhados.

Dica: use printf(“%*s[%s]\n”,level, “”,entry->dname) para imprimir com a identação correspondente ao nivel de recursão