Responsible lecturer: Illés, Zoltán associate professor
Media & Educational Informatics Department

File handling

Goal: We are going to get acquainted with file and directory system calls. We learn about error handling. We are to learn how to use binary files in C.

We learn about: *open, access, read, write, close – file system calls (include fcntl.h); O_RDWR, O_RDONLY, O_WRONLY, etc. – access modes; S_IWRITE,S_IREAD – permissions; lseek, SEEK_SET, SEEK_CUR, SEEK_END – repositions the offset of the file; opendir, readdir, closedir - directory system calls (include dirent.h); stat, fstat – functions resulting file status (include sys/types.h, sys/stat.h, unistd.h); ctime – convert t_time to string(include time.h); errno – error number; perror – writes out to the error output (include errno.h)*

Tasks

1. Write a C program which makes a copy of an optional file. The source file-name and the name of the copy has to be given by command line arguments! (*Open the source file as a binary one for reading and open the copy (destination file) for writing. Read the content of the original file character by character and write it out to the copy file!)*

```
…
int open(const char *path, int oflags);
int open(const char *path, int oflags, mode_t mode);
// path is the filename
// oflags: O_WRONLY - open for write, O_RONLY - open for read, O_RDWR - open
// for read and write, O_APPEND - open for append, O_TRUNC - open and delete
// the old content, O_CREATE-open a new file, O_EXCL - error if there was an old
//  mode - S_IRUSR, S_IWUSR, S_IXUSR,… reading, writing execution permissions
for the user

int  read(  int  handle,  void  *buffer,  int  nbyte );
// handle - file handler, buffer - address of variable to read in, nbyte - the
length of reading

int  write(  int  handle,  void  *buffer,  int  nbyte  );
// the same a sin read function

int  close(  int  handle  );
// handle - file descriptor
…
```

2. Write a C program which reads in some data from the keyboard and writes them out into a binary file! Create a struct for the data: name as character array and year of birth as integer. Write another C program which reads in the data from the above created file and writes it out on the screen! (*You have to use the same data structure in both of the programs so it is advised to make a separate file for it and include it into them.)*

3. Write a C program which lists the file-names of the actual directory!
You should open the actual directory, read the next file into a dirent structure from which you can get the file-name!

```
DIR *opendir(const char *name);
// name - directory path

struct dirent *readdir(DIR *dirp);

// dirp - directory descriptor
// result - struct dirent see below
/*
struct dirent {
```

Financed from the financial support ELTE won from the Higher Education Restructuring Fund of the Hungarian Government

```
        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 system types
        char           d_name[256]; // filename
    };*/
    ...
```

4. Modify the above written C program and write out the date of last modification of the files too! (*Use stat or fstat function and stat structure to decide the properties of the actual file!)*

```
int stat(const char *path, struct stat *buf);
// path – path of the file

// buf – the structure you can see below
/*struct stat {
    dev_t     st_dev;     // ID of device containing file
    ino_t     st_ino;     // inode number
    mode_t    st_mode;    // protection
    nlink_t   st_nlink;   // number of hard links
    uid_t     st_uid;     // user ID of owner
    gid_t     st_gid;     // group ID of owner
    dev_t     st_rdev;    // device ID (if special file)
    off_t     st_size;    // total size, in bytes
    blksize_t st_blksize; // blocksize for file system I/O
    blkcnt_t  st_blocks;  // number of 512B blocks allocated
    time_t    st_atime;   // time of last access
    time_t    st_mtime;   // time of last modification
    time_t    st_ctime;   // time of last status change
};*/
....
```

5. Modify the program to be able to list the content of the subdirectories as well! *(Check if the actual directory element is a directory or not.)*

...
int S_ISDIR(st_mode mode)
// mode - protectionmode
…

6. Try to write out the name of the owner as well like you see it in ls -al Unix command. (*You should use pwd.h include file, struct passwd and getpwuid functions!)*