3.5 格式化输入输出 与时间获取_物联网 / 嵌入式工程师 - 慕课网

> 慕课网慕课教程 3.5 格式化输入输出 与时间获取涵盖海量编程基础技术教程，以图文图表的形式，把晦涩难懂的编程专业用语，以通俗易懂的方式呈现给用户。

5.Linux 标准 io – 格式化输入输出 与时间获取

- 当遇到典型的格式化数据进行处理时, 就需要相应用于格式化输入 / 输出的函数来完成, 比如日期就是典型的具有格式的数据
    - 日期数据 : 2022 年 10 月 22 日
    - 地址数据 : 湖北省武汉市...
- 格式化输出函数如下:

函数头文件

#include <stdio.h>

函数功能

输出信息到标准输出

函数原型

int printf(const char *format, ...);

函数返回值

实际输出的字节数

函数头文件

#include <stdio.h>

函数功能

将格式化数据输出到标准输出

函数原型

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

函数参数

stream : 流对象指针

format : 格式字符串

函数返回值

实际输出的字节数

函数头文件

#include <stdio.h>

函数功能

将格式化数据输出到字符串缓冲区中

函数原型

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

函数参数

str : 字符串缓冲区地址

format : 格式字符串地址

函数头文件

#include <stdio.h>

函数功能

从标准输入读取格式化数据到缓冲区中

函数原型

int scanf(const char *format, ...);

函数参数

format : 格式字符串地址

函数返回值

实际读取的字节数

函数头文件

#include <stdio.h>

函数功能

从文件中读取格式化数据

函数原型

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

函数参数

str : 字符串缓冲区地址

format : 格式字符串地址

函数返回值

实际读取的字节数

函数头文件

#include <stdio.h>

函数功能

从字符串读取格式化数据

函数原型

int sscanf(const char *str, const char *format, ...);

函数参数

str：字符串地址

format：格式字符串地址

函数返回值

实际读取的字节数

```
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <string.h>
int main(int argc,char *argv[])
{
    FILE *fp = NULL;

    fp = fopen(argv[1],"w");
    if ( fp == NULL)
    {
        fprintf(stderr,"can't open file.\n");
        return -1;
    }

    int numa = 10;
    float numb = 1.23456;
    char *str = "Hello";
    char buffer[64];
    fprintf(fp,"%d-%f-%s",numa,numb,str);
    sprintf(buffer,"%d-%f-%s",numa,numb,str);

    puts(buffer);

    fclose(fp);
    return 0;
}
```

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


int main(int argc,char *argv[])
{
    FILE *fp = NULL;
    int numa = 0,numb = 0,numc = 0;
    char buffer[64] = "10-20-30";

    if (argc != 2){
        fprintf(stderr,"Usage : %s <pathname> \n",argv[0]);
        return -1;
    }

    fp = fopen(argv[1],"r");

    if (fp == NULL){
        perror("Error fopen(): ");
        return -1;
    }

    fscanf(fp,"%d-%d-%d",&numa,&numb,&numc);

    printf("numa = %d,numb = %d,numc = %d\n",numa,numb,numc);

    numa = 0,numb = 0,numc = 0;

    sscanf(buffer,"%d-%d-%d",&numa,&numb,&numc);

    printf("numa = %d,numb = %d,numc = %d\n",numa,numb,numc);



    fclose(fp);


    return 0;
```

```
}
```

- 在 Linux 中获取主要需要以下两个步骤
    - Step 1 : 通过 time() 函数获取从 1970 年至今的秒数
    - Step 2 : 通过 localtime() 或者 ctime() 函数

函数头文件

```
#include <time.h>
```

函数功能

- 获取从 1970-1-1 至今的时间秒数 (时间戳)

函数原型

```
time_t time(time_t *tloc);
```

函数参数

函数返回值

函数头文件

```
#include <time.h>
```

函数功能

- 将时间戳转换成本地时间, 并存储到 struct tm 结构体变量中

函数原型

```
time_t time(time_t *tloc);
```

函数参数

- struct tm *localtime(const time_t *timep);

函数返回值

- 返回 struct tm 结构体指针
  struct tm 结构体说明

```
struct tm {
 int tm_sec; /* Seconds (0-60) */
 int tm_min; /* Minutes (0-59) */
 int tm_hour; /* Hours (0-23) */
 int tm_mday; /* Day of the month (1-31) */
 int tm_mon; /* Month (0-11) */
 int tm_year; /* Year - 1900 */
 int tm_wday; /* Day of the week (0-6, Sunday = 0) */
 int tm_yday; /* Day in the year (0-365, 1 Jan = 0) */
 int tm_isdst; /* Daylight saving time */
```

};

获取当前时间并转换本地时间，以 %d-%d-%d %d::%d::%d 进行打印

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


int main(void)
{
        time_t t;

        struct tm *p_datetime;

        t = time(NULL);

        p_datetime = localtime(&t);

        printf(" %d-%d-%d %d::%d::%d\n"
                                        ,p_datetime->tm_year + 1900
                                        ,p_datetime->tm_mon + 1
                                        ,p_datetime->tm_mday
                                        ,p_datetime->tm_hour
                                        ,p_datetime->tm_min
                                        ,p_datetime->tm_sec );
        return 0;
}
```

练习：

- 获取系统时间，按照 <2022-5-8 23::15:00> 格式写入到文件中
-

全文完