دانشگاه بین المللی امام خمینی قزوین

دانشکدهٔ فنی و مهندسی

مبانی کامپیوتر و برنامه سازی

(دومین ترم کرونا!)

فصل چهاردهم: فایل ها در C

نستوه طاهری جوان

nastooh@aut.ac.ir

مقدمه

- محل نگهداری داده ها
  - در متغیرها (مانند آرایه ها و سایر ساختارها)
    - این متغیرها در RAM ساخته می شوند.
    - بعد از بسته شدن برنامه از بین می روند.

- سوال مهم:

اگر داده ها را برای مدت طولانی نیاز داشته باشیم چه باید کرد؟

nastooh@aut.ac.ir

معرفی فایل

- می توان داده های دائمی را در فایل ها ذخیره کرد.

- تعریف فایل:
  - محلی برای ذخیره و نگهداری داده ها بر روی رسانه های ذخیره سازی دائمی
    - مانند هارد دیسک، فلاپی، فلش، سی دی و ...
  - هر فایل حتما یک نام دارد.

nastooh@aut.ac.ir

انواع فایل

- انواع فایل در زبان C:
  - فایل متنی (Text File)
    - عموما برای ذخیرهٔ یک متن ASCII استفاده می شوند.
    - با ویرایشگرهای عادی مانند Notepad قابل خواندن هستند.
  - فایل دودویی (Binary File)

nastooh@aut.ac.ir

معرفی فایل های متنی

- خصوصیات فایل های متنی:
  - با ویرایشگرهای عادی مانند Notepad قابل خواندن هستند.
  - داده ها در قالب چندین خط نگهداری می شوند.
  - طول خط ها با هم برابر نیست.
  - کد ASCII هر کاراکتر ذخیره می شود.
  - انتهای هر خط \n وجود دارد.
    - « \n کد شمارهٔ ۱۰ اسکی است، یعنی 0xA
  - انتهای فایل با کاراکتر EOF مشخص می شود.
    - « کد شماره ۲۶، یعنی 0x1A

nastooh@aut.ac.ir

معرفی فایل های باینری

- خصوصیات فایل های باینری:
  - با ویرایشگرهای عادی نمی توان محتوای آنها را خواند.
  - داده ها به همان صورت حافظه، در دیسک ذخیره می شوند.
  - اندازهٔ خانه های فایل با هم برابر است.

- مثال: عدد ۲۳۶۱۷ در فایل متنی به صورت "23617" و در پنج بایت به صورت اسکی ذخیره می شود، اما در فایل باینری، همانطور که در حافظه ذخیره شده است، ذخیره خواهد شد.

nastooh@aut.ac.ir

اعمال روی فایل ها

- معمولا اعمالی مانند زیر روی فایل ها انجام می شود:
  - Read
  - Write
  - Update
  - Append

اما قبل از هر کاری، باید ابتدا فایل را باز کرد.

nastooh@aut.ac.ir

اعمال روی فایل ها

- باز کردن فایل
  - برای این کار از تابع fopen() استفاده می کنیم
    - الگوی این تابع در فایل sdtio.h قرار دارد.
  - برای بازکردن فایل ابتدا یک اشاره گر از نوع FILE نیاز داریم.
  - نحوهٔ استفاده از تابع fopen() به صورت زیر است:

```
FILE  *fp;
fp = fopen (char  *filename,  char  *mode);
```

که آرگومان اول نام فایل است و آرگومان دوم حالت باز کردن فایل را مشخص می کند.
که هر دو رشتهٔ کاراکتری هستند.

nastooh@aut.ac.ir

اعمال روی فایل ها

- باز کردن فایل
  - حالت های باز کردن فایل:

mode	کاربرد
r	خواندن از فایل متنی موجود
w	نوشتن در فایل متنی جدید
a	اضافه کردن به فایل متنی موجود
r+	خواندن و نوشتن از فایل متنی موجود
w+	خواندن و نوشتن در فایل متنی جدید
a+	اضافه کردن و خواندن از فایل متنی موجود

mode	کاربرد
rb	خواندن از فایل باینری موجود
wb	نوشتن در فایل باینری جدید
ab	اضافه کردن به فایل باینری موجود
rb+	خواندن و نوشتن از فایل باینری موجود
wb+	خواندن و نوشتن در فایل باینری جدید
ab+	اضافه کردن و خواندن از فایل باینری موجود

nastooh@aut.ac.ir

اعمال روی فایل ها

- باز کردن فایل
  - مثال برای باز کردن فایل:

```
FILE  *fp1, *fp2;
fp1 = fopen (“myfile.txt”,  “r+”);

fp2 = fopen (“myfile.dat”, “w+”);
```

- نکته مهم: اگر فایل های موجود را با حالت های w باز کنیم، محتوای قبلی از بین می روند. مراقب باشید!!!

nastooh@aut.ac.ir

اعمال روی فایل ها

- باز کردن فایل

- نکته مهم: اگر fopen() نتواند به هر دلیلی فایل را باز کند، NULL برمیگرداند. پس بهتر است مقدار خروجی تابع fopen() را حتما چک کنیم.

```
fp = fopen (“a.txt”, “r”);
if  ( fp = = NULL)
    printf (“File could not be opened”);
else
{
     do some things...
}
```


nastooh@aut.ac.ir

اعمال روی فایل ها

- بستن فایل
  - پس از اتمام پردازش مورد نظر در فایل، باید حتما فایل را بست.
  - با این کار اطلاعات بافر بر روی دیسک ذخیره می گردد.
  - برای بستن فایل از تابع fclose() استفاده می شود.
  - الگوی این تابع در فایل stdio.h قرار دارد.

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

- مثال:

```
fclose (fp);
```

- اگر عمل بستن با موفقیت انجام شود، این تابع عدد صفر و در غیر اینصورت عدد 1- را برمیگرداند.

nastooh@aut.ac.ir

اعمال روی فایل ها

- چند نکته در رابطه با فایل ها:
  - اگر فایلی برای خواندن باز می شود، باید از قبل وجود داشته باشد، و الا NULL برگشت داده می شود.
  - اگر فایلی که برای نوشتن باز می شود قبلا وجود داشته باشد، فایل قبلی از بین می رود!
  - در هنگام استفاده از حالت a اگر فایل از پیش موجود نباشد، ساخته می شود.
  - علامت + را در حالت، می توان وسط یا آخر گذاشت، مثلا w+b یا wb+.
  - هنگام نوشتن مسیر برای fopen، به جای \ باید از \\ استفاده کرد.

```
fp = fopen (“c:\\files\\myfolder\\a.txt”, “r+”);
```


nastooh@aut.ac.ir

اعمال روی فایل ها

- چند نکته در رابطه با فایل ها:
  - **نکته مهم:**
    - هر فایلی که باز می شود، یک اشاره گر داخلی دارد که مشخص می کند در هر لحظه خواندن یا نوشتن در کجای فایل باید انجام شود.
    - این اشاره گر بر اثر خواندن یا نوشتن به صورت اتوماتیک جلو می رود. ☺
    - هنگام باز کردن فایل با حالت های r و w این اشاره گر به ابتدای فایل اشاره می کند.
    - اما هنگام باز کردن فایل با حالت های a این اشاره گر به انتهای فایل اشاره می کند.

nastooh@aut.ac.ir

اعمال روی فایل ها

- خواندن و نوشتن در فایل
  - برای خواندن و نوشتن داده در فایل، روش های متنوعی وجود دارند، شامل توابع:
    - fgetc()
    - fputc()
    - fgets()
    - fputs()
    - fprintf()
    - fscanf()
    - fread()
    - fwrite()
    - ....

nastooh@aut.ac.ir

اعمال روی فایل ها

- خواندن کاراکتر از فایل متنی
  - برای این منظور از تابع fgetc() استفاده می کنیم.
  - الگوی آن در فایل stdio.h قرار دارد.
  - نحوه استفاده از آن به صورت زیر است:

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

  - این تابع کاراکتر خوانده شده از فایل را بر میگرداند.
  - این تابع اتوماتیک اشاره گر داخلی فایل را یکی به جلو می راند.
  - این تابع اگر به انتهای فایل برسد، EOF را برمیگرداند.

nastooh@aut.ac.ir

اعمال روی فایل ها

- خواندن کاراکتر از فایل متنی
  - مثال: *(محتوای فایل 01.txt در فولدر جاری را نمایش می دهد.)*

```
FILE *fp;
fp = fopen (“01.txt”, “r”);
if  (fp = = NULL){
    printf (“File could not be opened”);
}
else{
    do {
        ch = fgetc(fp);
        putchar (ch);
    } while (ch != EOF);
}
```


nastooh@aut.ac.ir

اعمال روی فایل ها

- نوشتن کاراکتر در فایل متنی
  - برای این منظور از تابع fputc() استفاده می کنیم
  - الگوی این تابع در فایل stdio.h قرار دارد.
  - نحوهٔ استفاده از آن به صورت زیر است:

```
int  fputc (char  ch, FILE  *fp);
```

  - این تابع کاراکتر مشخص شده را در فایل مورد نظر می نویسد.
  - تیم تابع اتوماتیک اشاره گر داخلی فایل را یکی به جلو می راند.
  - خروجی این تابع در صورت موفقیت، همان کاراکتر نوشته شده در فایل است. در غیر اینصورت مقدار EOF را برمیگرداند.

nastooh@aut.ac.ir

اعمال روی فایل ها

- خواندن رشته از فایل متنی
  - برای این منظور از تابع fgets() استفاده می کنیم.
  - الگوی این تابع در فایل stdio.h قرار دارد.
  - نحوهٔ استفاده:

```
fgets ( char  *s,  int length,  FILE  *fp);
```

  - پارامتر اول رشته ای است که پس از خواندن از فایل مقدار دهی می شود.
  - این تابع یک رشته را از یک خط از فایل می خواند، تا به انتهای خط برسد یا به اندازهٔ length خواندن را انجام می دهد.

nastooh@aut.ac.ir

اعمال روی فایل ها

- خواندن رشته از فایل متنی

  - مثال:

```
char  str[80];
FILE  *fp;
fp = fopen (“01.txt”, “r”);
fgets (str,  80,  fp);

puts (str);
```


nastooh@aut.ac.ir

اعمال روی فایل ها

- نوشتن رشته در فایل متنی
  - برای این منظور از تابع fputs() استفاده می کنیم.
  - الگوی این تابع در فایل stdio.h قرار دارد.
  - نحوهٔ استفاده:

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

  - پارامتر اول رشته ای است که باید در فایل نوشته شود.
  - مثال:

```
char  str[80];
FILE  *fp;
fp = fopen (“01.txt”, “w”);
gets(str);
fputs (str,  fp);
```


nastooh@aut.ac.ir

اعمال روی فایل ها

- نوشتن رشته در فایل متنی
  - در هنگام نوشتن رشته در فایل، در صورت نیاز \n را باید اضافه کرد.
  - مثال: (مقایسه کنید)

```
char  str[80];
FILE  *fp;
fp = fopen (“01.txt”, “w”);
gets(str);

fputs (str,  fp);
fputs (str,  fp);
fputs (str,  fp);
fputs (str,  fp);
```

```
char  str[80];
FILE  *fp;
fp = fopen (“01.txt”, “w”);
gets(str);

strcat (str , “\n”);

fputs (str,  fp);
fputs (str,  fp);
fputs (str,  fp);
fputs (str,  fp);
```


nastooh@aut.ac.ir

اعمال روی فایل ها

- تابع rewind()
  - از این تابع برای برگشت دادن اشاره گر داخلی فایل به ابتدای آن استفاده می کنیم.
  - الگوی آن در فایل sdtio.h قرار دارد.
  - نحوه استفاده:

```
rewind  (fp);
```

- تابع feof()
  - برای چک کردن اینکه آیا به انتهای فایل رسیده ایم یا خیر؟
  - نحوهٔ استفاده:

```
int  feof (fp);
```

  - اگر به انتهای فایل رسیده باشد، مقداری به جز صفر را بر میگرداند و اگر هنوز به انتهای فایل نرسیده باشد، مقدار صفر را برمیگرداند.

nastooh@aut.ac.ir

اعمال روی فایل ها

- مثال: خواندن فایل متنی و چاپ آن تا انتهای فایل

```
char s[80];
FILE *fp;

fp = fopen("0101.txt", "r");

fgets(s, 80, fp);
while (feof(fp) = = 0)
{
        printf("%s", s);
        fgets(s, 80, fp);
 }
```


nastooh@aut.ac.ir

اعمال روی فایل ها

- نوشتن در فایل با تابع fprintf()
  - از این تابع برای نوشتن فرمت بندی شده در فایل استفاده می کنیم
  - الگوی آن در stdio.h قرار دارد.
  - نحوهٔ استفاده:

```
fprintf (FILE  *fp,  رشتهٔ کنترلی ,  آرگومانها);
```

  - مثال:

```
fp = fopen("0101.txt", “w");
int  i=51;
float  f=34.54;
fprintf (fp, “%5d%.2%5d”, i, f, i);
```


nastooh@aut.ac.ir

اعمال روی فایل ها

- خواندن از فایل با تابع fscanf()

fscanf (FILE *fp, رشتهٔ کنترلی , آرگومانها);

- تمرین: آیا می توانید نحوهٔ استفاده از این تابع را مثال فرا بگیرید؟؟

nastooh@aut.ac.ir

اعمال روی فایل ها

- نوشتن در فایل با تابع fwrite()
  - از این تابع برای نوشتن داده در فایل استفاده می شود.
  - الگوی آن در فایل stdio.h قرار دارد.
  - توصیه می شود برای نوشتن ساختمان ها در فایل از این تابع استفاده شود.

fwrite (آدرس داده, اندازه داده , تعداد داده , اشاره گر فایل);

nastooh@aut.ac.ir

اعمال روی فایل ها

- نوشتن در فایل با تابع fwrite()
  - مثال:

```
struct  student{
...
...}  st;

fp = fopen("0101.txt", “w");

fwrite (&st, sizeof(struct student), 1, fp);
```


nastooh@aut.ac.ir

اعمال روی فایل ها

- خواندن از فایل با تابع ()fread
  - از این تابع برای خواندن داده از فایل استفاده می شود.
  - الگوی آن در فایل stdio.h قرار دارد.
  - نحوه استفاده:

fread (آدرس متغیر, اندازه داده, تعداد داده , اشاره گر فایل);

- توصیه می شود برای خواندن ساختمان ها از فایل از این تابع در کنار ()fwrite استفاده شود.

nastooh@aut.ac.ir

اعمال روی فایل ها

- خواندن از فایل با تابع fread()
  - مثال:

```
struct  student{
...
...}  st;

fp = fopen("0101.txt", “r");

fread (&st, sizeof(struct student), 1, fp);
```


nastooh@aut.ac.ir

ساختمان

- یادآوری چند نکته از scanf()

این نکات را هنگام دریافت فیلدهای ساختمان مد نظر قرار دهید.

- وقتی با تابع scanf() یک داده خوانده شود، کاراکتر اینترِ انتهای ورودی در بافر باقی خواهد ماند.
- اگر پس از scanf() بخواهیم یک رشته را با gets() بخوانیم، اینتر باقی مانده از scanf() را به عنوان ورودی gets() در نظر میگیرد و رشته را تهی مقدار دهی می کند.
- با scanf() نمی توان رشته های حاوی فضای خالی را خواند.
- می توان اعداد را نیز به صورت رشته ای خواند و با توابع کتابخانه ای مناسب، به مقدار عددی تبدیل کرد.

nastooh@aut.ac.ir

ساختمان

- تابع fseek()
  - این تابع برای انتقال اشاره گر داخلی فایل استفاده می شود.
  - الگوی آن در فایل stdio.h قرار دارد.
  - نحوهٔ استفاده از آن:

fseek (fp, میزان تغییر مکان , مبدا تغییر مکان);

  - این تابع اشاره گر داخلی فایل را از مقدار "مبدا تغییر مکان" به اندازهٔ میزان تغییر مکان جابجا می کند.
  - مبدا تغییر مکان می تواند مقادیر زیر باشد:

مقدار	مفهوم
SEEK_SET	ابتدای فایل
SEEK_CUR	مکان فعلی اشاره گر
SEEK_END	انتهای فایل

nastooh@aut.ac.ir

ساختمان

- تاملی در فایل باینری و متنی
  - اگر فایل فقط توسط برنامه (ماشین) قرار است باز شود، استفاده از فایل باینری بهتر است.
  - اما اگر انسان نیز قرار است به فایل دسترسی داشته باشد، (مثلا با برنامه notepad)، استفاده از فایل متنی مناسب تر است.
  - در عمل تفاوت بنیادی بین فایل باینری و متنی نیست. هر دو نوع فایل دنباله ای از صفر و یک ها هستند که بر روی رسانهٔ ذخیره سازی تحت یک نام ذخیره شده اند. فقط اینکه در مورد فایل متنی سعی می شود این صفر و یک ها را به صورت کدهای اسکی و بایت به بایت تفسیر کرد... همین!

nastooh@aut.ac.ir

پایان

nastooh@aut.ac.ir