File Access (7.5)

EECS 2031

12 November 2016

Declaring and Opening Files

```
FILE *fp; /* file pointer */
FILE *fopen( char *name, char *mode );
```

Example:

```
FILE *ifp, *ofp;
char iname[50], oname[50];
scanf( “%s %s”, iname, oname );
ifp = fopen( iname, "r" );
if ( ifp == NULL ) { ... }
ofp = fopen( oname, "w" );
if ( ofp == NULL ) { ... }
```

Modes

```
fp = fopen( name, "r" );
```

- Returns NULL if file does not exist, or has no read permission.

```
fp = fopen( name, "w" );
```

- If file does not exist, one will be created for writing.
- If file already exists, the content will be erased when the file is opened. So be careful!
- Returns NULL if file has no write permission.

3

Modes (cont.)

```
fp = fopen( name, "a" );    /* append */
```

- If file does not exist, one will be created for writing.
- If file already exists, the content will be preserved.
- Returns NULL if file has no write permission.

- May combine multiple modes.

```
fp = fopen( name, "rw" );
```

File may be read first, but the old content will be erased as soon as something is written to the file.

```
fp = fopen( name, "ra" );
fp = fopen( name, "aw" );  /* same as "a" */
```

4

Reading and Writing Files

```
int getc( FILE *fp )
int putc( int c, FILE *fp )
int fscanf( FILE *fp, char *format, ... )
int fprintf( FILE *fp, char *format, ... )

int c;
while ( (c = getc( ifp )) != EOF )
   putc( c, ofp );

char ch;
while ( fscanf(  ifp, "%c", &ch ) != EOF )
   fprintf(  ofp, "%c", ch );
```

5

Closing Files

```
int fclose( FILE *fp )

fclose( ifp );
fclose( ofp );
```

- Most operating systems have some limit on the number of files that a program may have open simultaneously ⇒ free the file pointers when they are no longer needed.
- **`fclose`** is called automatically for each open file when a program terminates normally.
- For output files: **`fclose`** flushes the buffer in which **`putc`** is collecting output.

6

Reminder: I/O Redirection

- In many cases, I/O redirection is simpler than using file pointers.

```
a.out < input_file > outout_file

a.out < input_file >> outout_file
```

7

Macro Substitution (4.11.2)

- #define *name* *replacement_text*
  - subsequent occurrences of the token name will be replaced by the *replacement text.*
- Macro substitutions are faster than function calls.

```
#define max(A, B) ((A) > (B) ? (A) : (B))
x = max( p+q, r+s );
/* x = ( (p+q) > (r+s) ? (p+q) : (r+s) ); */
i = 1; j = 10;
y = max( i++, j++ );  /* final values of i and j ?*/

#define square(x) x * x   /* what's wrong? */
z = square(y + 1);
```

8