

```
#define: Macros

Use parenthesis, as otherwise
#define CIRCLE_AREA( x ) PI * x * x
area = CIRCLE_AREA( c + 2 );
becomes
area = 3.14159 * c + 2 * c + 2;

Evaluates incorrectly

Macro's advantage

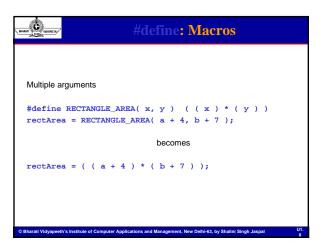
Avoid function call overhead

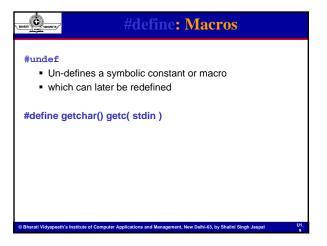
As, macro inserts code directly.

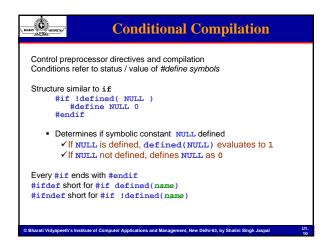
Macro's disadvantage

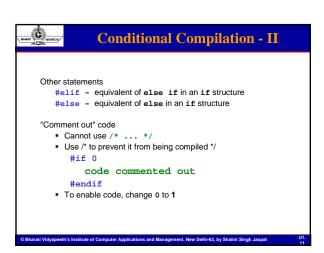
Argument may be evaluated more than once.

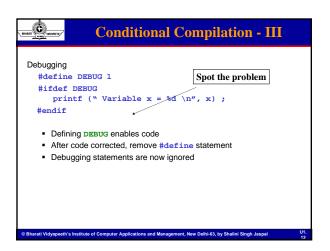
While defining, don't give a space between Macro Name & parenthesis. Why!!!
```

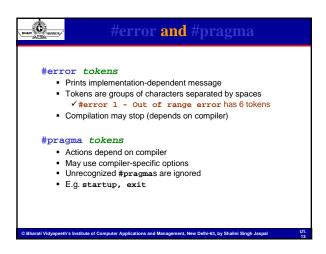


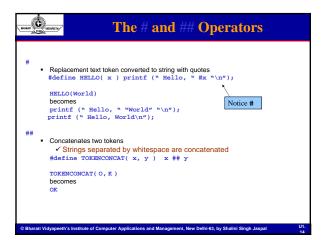


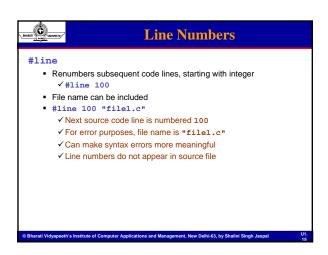














Predefined Symbolic Constants

Five predefined symbolic constants

■ Cannot be used in #define or #undef

Symbolic constant	Description
LINE	The line number of the current source code line (an integer constant).
FILE	The presumed name of the source file (a string).
DATE	The date the source file is compiled (a string of the form "Mmm dd yyyy" such as "Jan 19 2001").
TIME	The time the source file is compiled (a string literal of the form "hh:mm:ss").



Assertions

assert macro

- Header <assert.h>
 Tests value of an expression
 If 0 (false) prints error message and calls abort

assert(x <= 10);

If NDEBUG defined...

- Define before including assert.h
 All subsequent assert statements ignored
 #define NDEBUG



LIBRARY FUNCTIONS IN **C LANGUAGE**

Name of the second of the seco
Library functions is a set of ready-made software routines (functions) for programmers.
❖To use a library function, it is necessary to call the appropriate header file at the beginning of the program.
The C language is accompanied by a number of standard library functions which carry out various useful tasks. Some of them are discussed as follows:
© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Dethi-63, by Shalini Singh Jaspal
Name of the Park o

SOME GENERAL FUNCTIONS

<STDLIB.H>

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Shalini Singh Jaspal



bsearch

 $\ \, \underline{ PURPOSE}:$

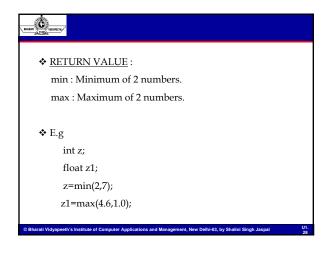
Performs the binary search operation.

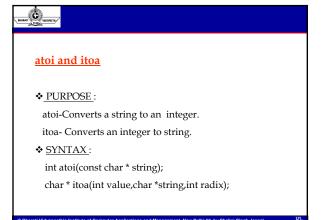
❖ <u>SYNTAX</u> :

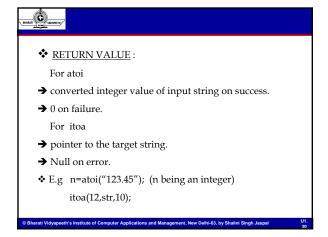
© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Shalini Singh Jaspal

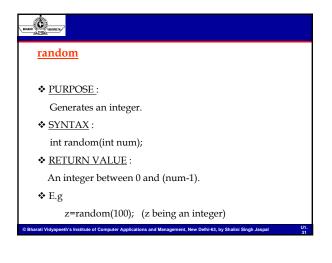
MANAGE RESERVE
❖ PARAMETER DESCRIPTION:
→ key : Value to be searched.
→ base : 0 th element of the array.
→ nelem: Number of elements in the array.
→ width: Number of bytes in each entry.
→ fcmp(): A user defined comparison routine
that compares 2 elements and returns
a value based on the comparison.
 Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Shalini Singh Jaspal
Manage Manage
❖ RETURN VALUE:
The address of the first entry in the array that
matches the key , on success. 0 , on failure (No match).
• E.g
* E.g int *itemptr;
itemptr = (int *) bsearch (&key, numarray,
NELEMS(numarray), sizeof(int),
numeric);
numenc _j ,
© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Shalini Singh Jaspal
NAME OF THE PROPERTY.
<u>qsort</u>
❖ PURPOSE :
Sorts an array using Quick sort algorithm.
~ · · · · · · · · · · · · · · · · · · ·
❖ SYNTAX:
void qsort(void *base, size_t nelem, size_t width,
int (*fcmp)(const void *, const void *));
in (temp/(const void , const void))),

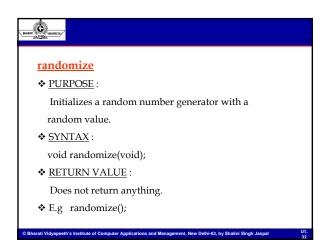
Name of State of Stat	
❖ <u>PARAMETER DESCRIPTION</u> :	
→base : 0 th element of the array.	
→ nelem : Size of the array.	
→ width: Number of bytes in each entry.	
→ fcmp(): A user defined comparison routine	-
that compares 2 elements and returns	
a value based on the comparison.	
© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Shalini Singh Jaspal UI. 25	
akt.	•
No. of Contract of	
❖ <u>RETURN VALUE</u> :	
	-
Does not return anything.	
Å E a	
sizeof(int), numeric);	
© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Deihi-63, by Shalini Singh Jaspal 06.	
28	
New Comment	
Various Acad Related A	
min and max	
* DUDDOCE	
* PURPOSE:	
min : Finds the smaller of 2 values.	
max : Finds the larger of 2 values.	
4. CVATTAY	
* SYNTAX:	
(type) min(a,b);	
(type) max(a,b);	
© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Shalini Singh Jaspal U1. 27	

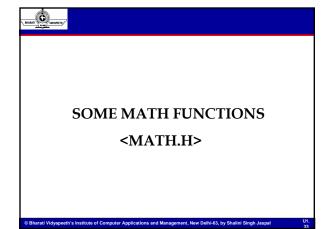


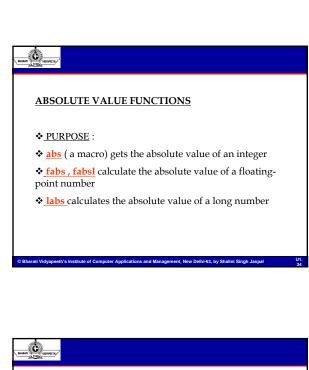


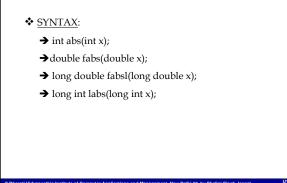


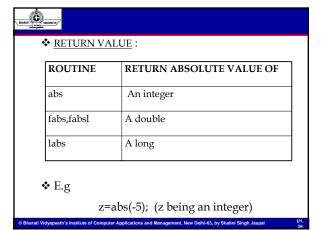










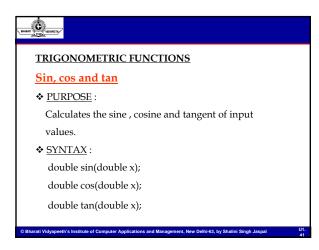


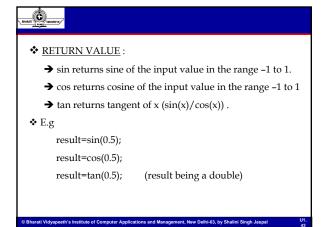
MADE TO THE PROPERTY OF THE PR
EXPONENTIAL FUNCTION
<u>exp</u>
❖ <u>PURPOSE</u> :
Calculates e raised to the power xth value.
❖ <u>SYNTAX</u> :
double exp(double x);
❖ <u>RETURN VALUE</u> :
e ^x value , given x
❖ E.g result=exp(10.3); (result being a double)
© Bharati Vidyapeeth's institute of Computer Applications and Management, New Delhi-53, by Shalini Singh Jaspal
w boardu vioyapeem's institute of Computer Applications and Management, New Delmi-6s, by Shalini Singh Jaspai
allo.
V SOUND OF THE PARTY OF THE PAR
LOGARITHMIC FUNCTION
log and log10
❖ PURPOSE:
log: Calculates natural logarithm of double var.
log10: Calculates base 10 logarithm of double var
❖ SYNTAX:
double log(double x);
double log10(double x);
double logio(double x),
© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Shalini Singh Jaspal
NAME OF THE PROPERTY OF THE PR
♣ DETUDNIVALUE
❖ <u>RETURN VALUE</u> :
log: Natural log of x.
log10 : Log (base 10) of x.
❖ E.g

result=log(10.3);

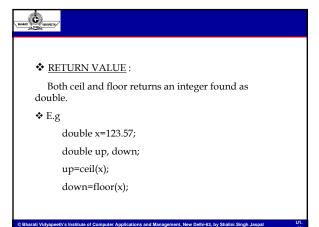
result=log10(10.3); (result being a double)

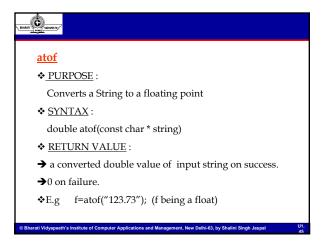
· mune Lander	
SQUARE ROOT FUNCTION	
<u>sqrt</u>	
❖ <u>PURPOSE</u> :	
Calculates square root of a variable.	
❖ <u>SYNTAX</u> :	
double sqrt(double x);	
❖ <u>RETURN VALUE</u> :	
Square root of x , if x is real and positive.	
E.g result=sqrt(4);	
© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Shalini Singh Jaspal	U1. 40





Tourse Transmit	
BOUNDARY FUNCTIONS	
Ceil and floor	
❖ PURPOSE:	
ceil: Finds the smallest integer not < its argument.	
floor: Finds the largest integer not > its argument.	
❖ <u>SYNTAX</u> :	
double ceil(double x);	
double floor(double x);	
	U1.
© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Shalini Singh Jaspal	43





BARRET TO STREET STREET		
:	SOME TIMING FUNCTIONS	

<TIME.H>



<u>time</u>

 \diamond <u>PURPOSE</u>:

Gets the time of the day in seconds.

❖ <u>SYNTAX</u>:

time_t time(time_t *timer);

time_t is a arithmetic type capable of representing time. However, how time is encoded within this type is unspecified.

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Shalini Singh Jaspal

47



❖ <u>RETURN VALUE</u> :

The current time, in seconds, elapsed since 00:00:00 GMT, January 1, 1970.

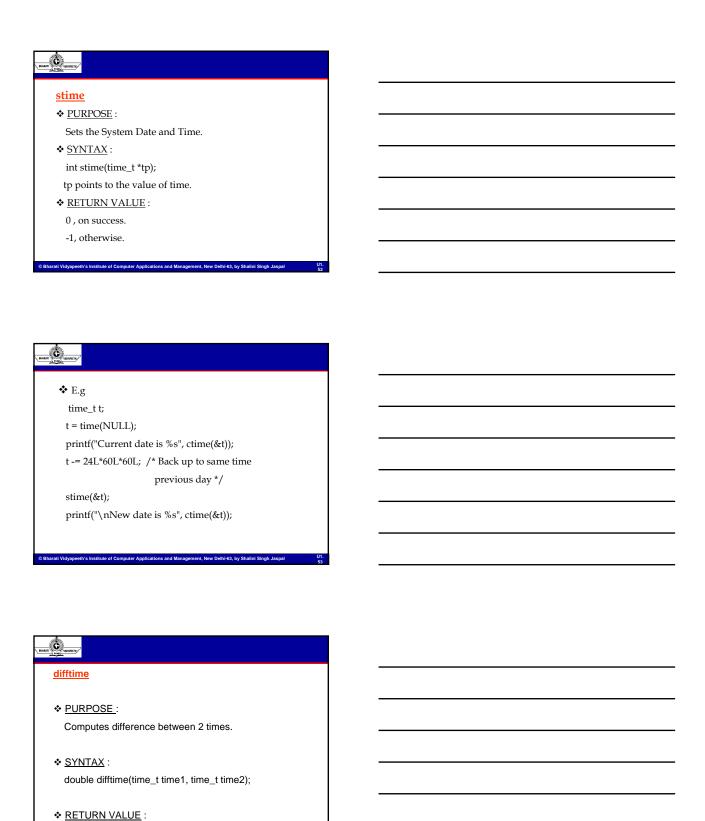
Stores the returned value in *timer variable.

❖ E.g

time_t t; t=time(NULL); //(or time(&t)) printf("The number of seconds since January 1, 1970 is %ld",t);

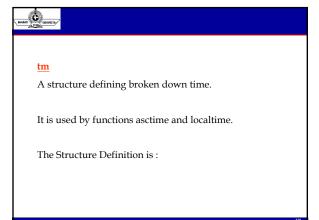
Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Shalini Singh Jasp

NAME OF THE PROPERTY.	
<u>ctime</u>	
❖ <u>PURPOSE</u> :	
Converts Date and Time to a string.	
❖ SYNTAX:	
<pre>char * ctime(const time_t * time);</pre>	
It converts the time value * time into a	
26-character long string.	
© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Shalini Singh Jaspal VI.	
49	
(\$	
A STATE OF THE PARTY OF THE PAR	
The 26-character string is in the following form	
and is terminated with a \n and $\0$ character:	
DDD MMM dd hh : mm : ss YYYY	
where: DDD = Day (Mon, Tue, Wed, etc.)	
MMM = Month (Jan, Feb, Mar, etc.)	
dd = Date (1, 2,, 31) hh = Hour (1, 2,, 24)	
mm = Minutes (1,, 59) ss = Seconds (1,, 59)	
YYYY = Year	_
© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Dethi-63, by Shalini Singh Jaspal U1.	
Camara,	
❖ RETURN VALUE:	
Returns a pointer to the character	
string containing the date and time.	
❖ E.g time_t t;	
time(&t);	
printf("Today's date and time: %s\n", ctime(&t));	
Tues Nov 4 11:31:54 2008\n\0	
© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Shalini Singh Jaspal 51.	



Returns the elapsed time in seconds.

```
* E.g:
time_t first, second;
first = time(NULL); /* Gets system
time */
delay(2000); /* Waits 2 secs */
second = time(NULL); /* Gets system time
again */
printf("The difference is: %f seconds\n",
difftime(second,first));
```



```
struct tm {

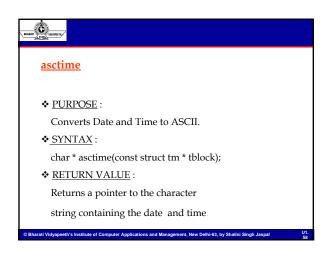
int tm_sec;    /* Seconds */
int tm_min;    /* Minutes */
int tm_hour;    /* Hour (0-23) */
int tm_hour;    /* Day of month (1--31) */
int tm_mon;    /* Month (0--11) */
int tm_year;    /* Year (calendar year minus 1900) */
int tm_wday;    /* Weekday (0--6; Sunday = 0) */
int tm_yday;    /* Day of year (0--365) */
int tm_isdst;    /* 0 if daylight savings time is not in

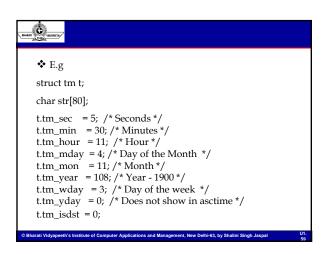
effect */
};

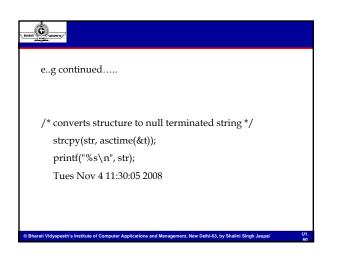
**C **Bharati Vidyapeeth's institute of Computer Applications and Management, New Dehi-13, by Shalini Singh Jaspal

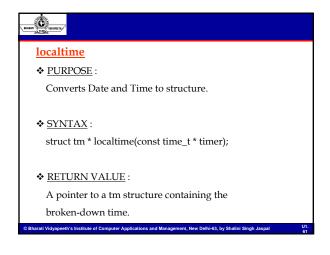
**U.**
**C **Bharati Vidyapeeth's institute of Computer Applications and Management, New Dehi-13, by Shalini Singh Jaspal

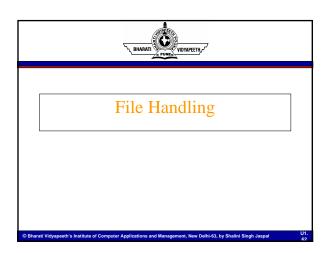
**U.**
**U.**
**C **Bharati Vidyapeeth's institute of Computer Applications and Management, New Dehi-13, by Shalini Singh Jaspal
```

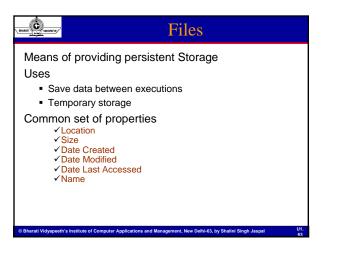














Need For Files

All the programs we worked on so far

- Get their input from the keyboard.
- Write output to the screen.

Works well if

- Input data size is small / data is being entered for the first time.
- Data entered / Output generated is not needed after the program execution is over.

Requires test data / actual data to be entered every time the program is run.

Files can be are used in order to overcome these problems.

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Shalini Singh Jaspal

U1.



Files in C

- A file must first be opened properly before it can be accessed for reading or writing.
- When a file is opened, a stream is associated with the file.
- Successfully opening a file returns a pointer to (i.e., the address of) a file structure.
- This pointer contains the information required to control the file at runtime.

Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Shalini Singh Jaspai

U1.



I/O Streams in C

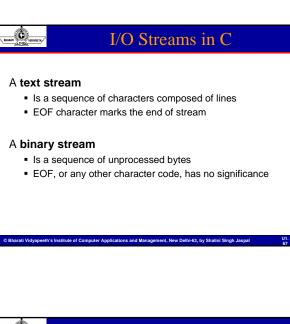
File handling in C is provided by a number of functions in the standard library

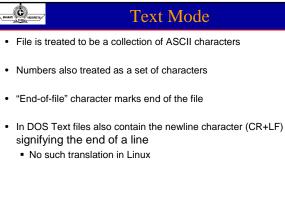
All input/output is based on the concept of a stream

There are two types of streams

- Text stream
- Binary stream

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Shalini Singh Jaspa





@ Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Shalini Singh Jaspal



Binary Mode

- File is treated to be a collection of binary numbers
- No special EOF character
- No special treatment of newline characters
- Numbers stored as corresponding binary equivalents

Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Shalini Singh Jaspa

Standard Streams in C Description Name Remark stdin Standard Input Connected to the Stream Keyboard Standard Output Connected to the Screen stdout Stream Standard Error Connected to the Screen stderr Stream Not redirectable

- Others: stdprn, stdaux
- The compiler produces code to open these files before the main function
- is invoked.

 The I/O macros and functions we have used so far read from stdin and write to stdout.
- These include getchar, putchar, scanf, and printf



File Processing



File Processing

The basic steps involved in processing data contained in files are:

- Open / Create the file
- Process the data ✓ Write / Read / Append
- Close the Files

A failure to close the file after processing is over may lead to data corruption.



Opening a File

- When a computer program opens a file, no data is read from it.
- The open request is given to the operating system.
- The operating system finds where the file is stored and issues a "file handle".
- When the program wants to read or write to the file at this point, it uses this file handle (the file descriptor) as a parameter to generate operating system requests.
- When a file is opened, the logical name is translated into a physical location on a specific device (e.g. a disk drive)

Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Shalini Singh Jaspal

U1.



The FILE ADT

A struct type defined in stdio.h

Represents information about a file stream

Required to work on the file data

In general, the function fopen() prepares a file for use

Normally accessed via

- Pointers of FILE * type,
- Created using fopen()
- A function declared in the standard header: stdio.h

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Shalini Singh Jaspal

U1



fopen()

fopen(file_name, mode)

opens the named file in a particular mode and returns a file pointer.

FILE *fopen(char *file_id, char *mode);

- file_id is the path, file name, and extension
- mode is how we intend to use it
- The return value is the file pointer: Use it in all subsequent references to the file.
- If fopen fails, NULL is returned.

FILE *payfile = fopen("payroll.txt", "r");

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Shalini Singh Jaspal

Mode	Operations Allowed	Action	
r	Read	Return NULL if file doesn't exist	Mode is passed as a string
w	Write	Create if file doesn't exist. Destroy file contents if it exists	constant / variable
а	Write at end	Create if file doesn't exist. Retain old contents	I DOUBLE OF
r+	Read Write	Return NULL if file doesn't exist	In DOS Use t / I in order to specify text /
W+	Read Write	Create if file doesn't exist. Destroy file contents if it exists	binary mode
a+	Read,	Create if file doesn't exist.]
	Write-at-end	Retain old contents	



The exit() function

- Causes normal program termination.
- Sometimes error means we want an "emergency exit" from an application.
- In main we can use "return" to return to OS.
- In other functions we can use exit to do this.
- Exit is declared in the stdlib.h header

exit(1); in a function is exactly the same as
return 1; in the main routine

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Shalini Singh Jaspal

77



The exit () Function

- This is used to leave the program at anytime, from anywhere, before the "normal" exit location.
- Syntax:
 - exit (status);
- Example:

if((fp=fopen("a.txt","r")) == NULL){
 fprintf (stderr, "Cannot open a.txt!\n");
 exit(1);
}

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Shalini Singh Jaspal



Four Ways to Read and Write Files

- · Get and put a character
- · Get and put a line
- Formatted file I/O
- · Block read and write

® Rharati Vidyanaeth's Institute of Computer Applications and Management, New Delhi-63, by Shalini Singh, Issnal

U1.



Read / Write a Character

```
#include <stdio.h>
int fgetc(FILE * fp);
int fputc(int c, FILE * fp);
```

- These two functions read or write a single byte from or to a file.
- fgetc returns the character that was read, converted to an integer.
- fputc returns the same value of parameter c if it succeeds, otherwise, return EOF.

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Shalini Singh Jaspal





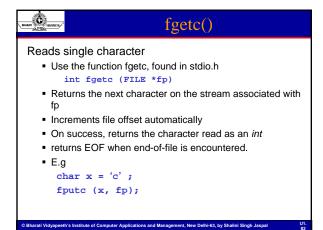
fputc()

Writing single character

- Use the function fputc, found in stdio.h
 int fputc (int c, FILE *fp)
- Writes the next character to stream associated with fp
- Increments file offset automatically
- On success, returns integer value of characters written
- On failure, returns EOF
- E.g

char x = 'c';
fputc (x, fp);

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Shalini Singh Jaspal

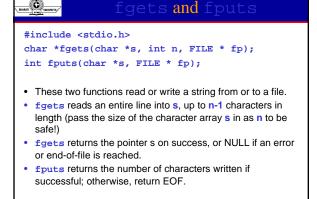




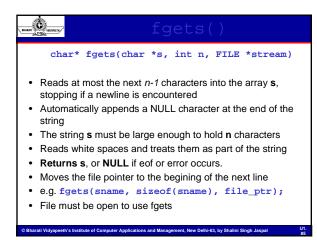
Reading / Writing Strings

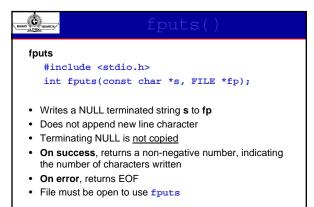
Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Shalini Singh Jaspal

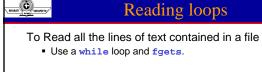
83

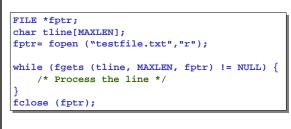


© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Shalini Singh Jaspal





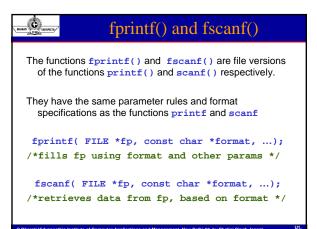




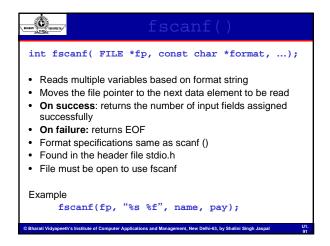
© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Shalini Singh Jaspal

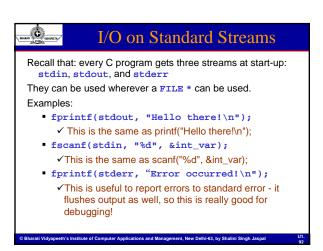


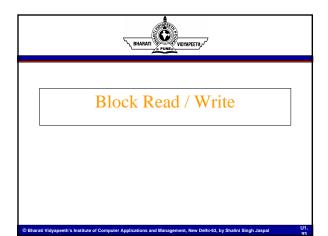
Formatted I/O



int fprintf(FILE *fp, const char *format, ...); • Use function fprintf for multiple variable writes • On success returns number of bytes written • On failure returns EOF • Format specifications same as printf() • Found in the header file stdio.h • File must be open to use fprintf Example fprintf(fp, "Name:%s, Pay:%.2f\n", name, pay);









fread and fwrite

Reading and Writing arrays of bytes

- Use the functions fread and fwrite
- Found in the header file stdio.h

Both of them takes four parameters

- a pointer to the array's first element (void*)
- the size (in bytes) of an element
- the number of elements in the array
- a file pointer

U1.



Generic Form of fwrite / fread

```
int fread / fwrite
  (void *buf, int size, int count, FILE *fp);
```

Examples

 Writing an array of integers arr to the file associated with fp

fwrite(arr, sizeof(int), 100, fp);

- Reading from a file associated with fp into an array arr fread(arr, sizeof(int), 100, fp);
- fwrite (fread) returns the number of items actually written (read).

© Bharati Vidyapeeth's institute of Computer Applications and Management, New Delhi-63, by Shalini Singh Jaspai

U1.

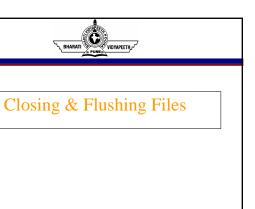


Reading / Writing Composite Variables

```
struct student stud;
/* Populate stud */
fwrite( &stud, sizeof(stud), 1, fp);
```

struct student studList[MAX];
/* Populate stud list, say count entries */
fwrite(studList, sizeof(stud)*count, 1, fp);

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Shalini Singh Jaspal





Closing a File

When a program has no further use for a file, it should close the file

Flushes the buffers associated with the file

int fclose(FILE * fp;

Return Value:

- On Success: 0
- On Failure: EOF

Files are automatically closed at normal program termination, including exit()

 They are not closed when the program crashes and Vidyapeeth's institute of Computer Applications and Management, New Delhi-53, by Shalini Singh Jaspal

U1.



Flushing Files

- Buffer can be cleared without closing, if required
 - int fflush (FILE * fp) ;
- Essentially this is a force to disk.
- · Very useful when debugging.
- Without fclose or fflush, updates to a file may not be written to the file on disk.
- Operating systems like Unix usually use write caching disk access.

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Shalini Singh Jaspal







Sequential / Random Access

There are two ways of accessing a file, either

- Sequential access
 - ✓ Accessing one data element after another in a file
 - ✓ If the data is to be process in the order of storage
 - ✓ Generally used if all / most of the data records in a given range are to be processed
- Random/direct access
 - ✓ Accessing characters at any position in a file
 - ✓ Can randomly decide where to start reading/writing a
 file.
 - ✓Use fseek and ftell

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Deihi-63, by Shalini Singh Jaspal





Sequential Access

- The OS maintains data to indicate the position of your next reading or writing.
- When you read/write, the position moves forward.
- You can "rewind" and start reading from the beginning of the file again:
 - void rewind (FILE * fp);

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Shalini Singh Jaspal



To determine where the position indicator is use:

✓ Returns a long giving the current position in bytes.

long pos= ftell (fp) ;

✓The first byte of the file is byte 0.

✓If an error occurs, ftell () returns -1.

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Shalini Singh Jaspal



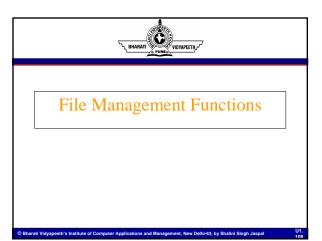
Detecting End-Of-File

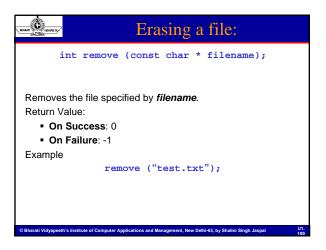
© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Shalini Singh Jaspal

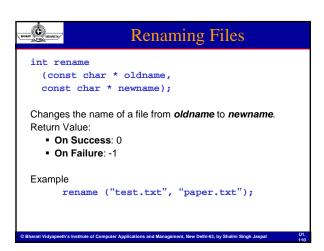
•

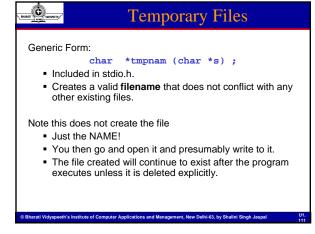
Text Mode while ((c = fgetc (fp)) != EOF) Reads characters until it encounters the EOF The problem is that in binary files the byte of data read may actually be indistinguishable from EOF.

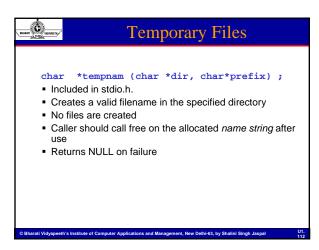
int feof (FILE * fp); The feof function realizes the end of file only after a reading failed (fread, fscanf, fgetc ...) fseek(fp,0,SEEK_END); printf("%d\n", feof(fp)); /* zero */ fgetc(fp); /* fgetc returns -1 */ printf("%d\n",feof(fp)); /* nonzero */













Temporary Files

Files that only exist during the execution of the program. Generic Form:

FILE *tmpfile (void) ;

- Included in stdio.h.
- Creates a temporary file in w+b mode
- returns FILE * on success
- returns NULL on failure
- The file is automatically removed when it is closed or when program terminates.

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Deihi-63, by Shalini Singh Jaspal