The University Of Sheffield.

C PROGRAMMING

- Characters and Strings
- File Processing
- Exercise

The University Of Sheffield.

CHARACTERS AND STRINGS

- A single character defined using the char variable type
- Character constant is an int value enclosed by single quotes
  - E.g. ‘a’ represents the integer value of the character a
- A string is a series of characters
  - String, string literals and string constants enclosed by double quotes

The University Of Sheffield.

DEFINING CHARACTERS AND STRINGS

- Declaring and assigning a single character

  char c='a';

- Strings are arrays of characters
  - A pointer to the first character in the array
  - The last element of the string character array is the null termination charcter '\0'
  - '\0' Denotes theend of a string

DEFINING STRINGS

- char node[]="iceberg";
- char *nodeptr="iceberg";
- char nodename[180];
- For the first two definitions the null termination is added by the compiler

The University Of Sheffield.

CHARACTER INPUT AND OUTPUT

- include <stdio.h>
- int getchar(void)
  - Input the next character from standard input, return it as an integer.
- int putchar(int c)
  - Display character stored in c
- Also use printf and scanf with the %c format specifier

The University Of Sheffield.

STRING INPUT AND OUTPUT

- char *gets(char *s)
  - Input characters from standard inout in to the array s until newline or EOF character is reached. A NULL termination character is placed at the end of the string.
- int puts(char *s)
  - Display array of characters in s follow with a newline.
- Also use printf and scanf with the %s format specifier

The University Of Sheffield.

CODE EXAMPLE USING PUTS AND GETCHAR

```
char c, nodename1[80], nodename2[80];
int i=0;

puts("Enter a line of text");
while((c=getchar())!='\n')
    nodename1[i++]=c;
nodename1[i]='\0';
```

The University Of Sheffield.

FORMATTED STRING INPUT AND OUTPUT

- sprintf(char *s, const char *format, …..)
  - Equivalent to printf with the exception that its output is stored in the array s specified in the sprintf function. The prototype for sscanf is ;

- sscanf(char *s, const char *format, …).
  - Equivalent to scanf reads input from the string s specified in the sscanf function.

The University Of Sheffield.

SPRINTF AND SSCANF EXAMPLES

```
char node[20], s2[80];
char s1[] ="Titania 3.78 7";
float fload, floadout;
int nusers, nusersout;

/*Using sscanf to read data from a string*/
sscanf(s1, "%s%f%d", node, &floadout, &nusersout);
sprintf(s2, "%s %f %d", node, fload, nusers);
```

The University Of Sheffield.

FUNCTIONS FOR CHARACTER MANIPULATION

- library **ctype.h**
- **isdigit**, **isalpha**, **islower**, **isupper**, **toupper**, **tolower** and **isspace**.
- These functions can be used to perform conversions on a single character or for testing that a character is of a certain type.

STRING CONVERSION FUNCTIONS

- String conversion functions from the general utilities library **stdlib**
- convert strings to float, int long int, double, long, and unsigned long data types respectively.
- **atof**, **atoi**, **atol**, **strtod**, **strtol**, **strtoul**

The University Of Sheffield.

STRING MANIPULATION

- The string handling library **string.h**
- provides a range of string manipulation functions for copying, concatenating, comparison, tokenizing and for identifying the occurrence and positions of particular characters in a string.
- E.g. **strcpy**, **strlen**, **strcmp** and **strtok**.
- See the examples

The University Of Sheffield.

FILE PROCESSING

- file as a sequential stream of bytes with each file terminated by an end-of file marker
- When a file is opened a stream is associated with the file
- Streams open during program execution
  - stdin
  - stdout
  - stderr

The University Of Sheffield.

SEQUENTIAL FILE MANAGEMENT

- Streams
  - channels of communication between files and programs.
- Range of functions for streaming data to files
  - fprintf
  - fscanf
  - fgetc
  - fputc

The University Of Sheffield.

OPENING A FILE

- When opening a file it is necessary to declare a variable that will be used to reference that file, the standard library provides the FILE structure.
- So a pointer to a FILE is declared using:
  - FILE *myfile;
- File opened using the function fopen
  - returns a pointer to the opened file

The University Of Sheffield.

FOPEN USAGE

```
if((myfile=fopen("myfilename", "w"))==NULL)
                        printf("The file could not be opened!\n");
else
{
                file was opened and is read or written here
}
```

The University Of Sheffield.

FILE OPEN MODES

Mode	Description
r	Open for reading
w	Open for writing
a	Append, open or create a file for writing at the end of the file
r+	Open a file for update (reading and writing)
w+	Create a file for update. If the file already exists discard the contents
a+	Append, open or create a file for update, writing is done at the end of the file

WRITING DATA USING FPRINTF

- *fprintf(fileptr, "format specifiers", data list);*
  - fprintf(mfptr, "%6d %20s %6d\n", iRunid, sName, iNode);
- Closing the file
  - fclose(mfptr);

READING DATA USING FSCANF

- *fscanf(fileptr, "format specifiers", data list);*

```
while(!feof(mfptr))
{
        printf("%6d %20s %6d\n", sim.id, sim.name, sim.node);
        fscanf(mfptr, "%d%s%d", &sim.id, sim.name, &sim.node);
}
```

The University Of Sheffield.

PRACTICAL CODING EXAMPLE

- Method for solving 1st order ODEs with well defined BC's
- Shooting Method
  - Compile and run the code
  - startshooting.c

The University Of Sheffield.

EXERCISE

- Adapt the program startshooting.c to read the input parameters from an input file.
- Adapt the program so that it reads the guess q froom the command line
- To read parameters from the command line we use the parameters argc and argv passed into the main function
- Use the following line to convert the command line parameter
  - Hint look at vecdp.c in the functions folder
  - If(argc>1)

```
q=atof(argv[1]);
```

The University Of Sheffield.

RANDOM ACCESS FILES

- Transaction processing systems
- Individual records of same length accessed at random
- fwrite
  - Write fixed number of bytes to a file
- fread
  - Read fixed number of bytes from a file

The University Of Sheffield.

DATA DECLARATION

- Example data structure
  - struct mydata

    { int index; float data;}
- Typical declaration
  - struct mydata blankdata={0, 3.141};

FWRITE – EXAMPLE CALL

- fwrite(&blankdata, sizeof(struct mydata), 1, fileptr)
  - Write data structure myblankdata
  - Specify correct field size
  - Specify number of data items to write (in this case 1)
  - Provide a valid pointer to the file that is opened for writing

FREAD – EXAMPLE CALL

- fread(&blankdata, sizeof(struct mydata), 1, fileptr)
  - Read data structure myblankdata
  - Specify correct field size
  - Specify number of data items to read (in this case 1)
  - Provide a valid pointer to the file that is opened for reading

The University Of Sheffield.

FSEEK

- Fseek sets file pointer to specific position in file

- int fseek(FILE *stream, long int offset, int whence)
- Offset is number of bytes from location whence
- Whence has one of three values
  - SEEK_SET (beginning of file)
  - SEEK_CUR (current location)
  - SEEK_END (end of file)
- Example call

  - fseek(myfileptr, sizeof(struct mydata)*(index-1),SEEK_SET);

The University Of Sheffield.

EXAMPLE

- Study and run the program fileio.c in the extras directory