j0() — Mathematics Function (libm)

Compute Bessel function
#include <math.h>
double j0(*z*)
double *z*;

j0() computes the Bessel function of the first kind for order 0 for its argument *z*.

Example

This example, called **bessel.c**, demonstrates the Bessel functions **j0()**, **j1()**, and **jn()**. Compile it with the following command line

```
      cc -f bessel.c -lm
```

to include floating-point functions and the mathematics library.

```
#include <errno.h>
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#define display(x) dodisplay((double)(x), #x)

dodisplay(value, name)
double value; char *name;
{
      if (errno)
            perror(name);

      else
            printf("%10g %s\n", value, name);
      errno = 0;
}

main()
{
      extern char *gets();
      double x;
      char string[64];

      for(;;) {
            printf("Enter number: ");
            if(gets(string) == NULL)
                  break;
            x = atof(string);

            display(x);
            display(j0(x));
            display(j1(x));
            display(jn(0,x));

            display(jn(1,x));
            display(jn(2,x));
            display(jn(3,x));
      }
}
```

See Also

j1(), jn(), libm

j1() — Mathematics Function (libm)

Compute Bessel function
#include <math.h>
double j1(*z***) double** *z***;**

j1() takes *z* and computes the Bessel function of the first kind for order 1.

Example

For an example of this function, see the entry for **j0()**.

See Also

j0(), jn(), libm

jn() — Mathematics Function (libm)

Compute Bessel function
#include <math.h>
double jn(*n***,** *z***) int** *n***; double** *z***;**

jn() takes *z* and computes the Bessel function of the first kind for order *n*.

Example

For an example of this function, see the entry for **j0()**.

See Also

j0(), j1(), libm

jobs — Command

Print information about jobs
jobs

The command **jobs** is used with the Korn shell's job-control feature. It prints information about all background jobs. The information printed is in the following format:

> **%***num* [**+-**] *pid status command*

num indicates the job number, **+** indicates that the job is the "current job"; **-** indicates that it is the "previous job". *pid* gives the process identifier of the job. *status* indicates the status of the job. *command* gives the job's command line.

For details about job control, see the Lexicon entry for **ksh**.

See Also

commands, ksh

join — Command

Join two data bases
join [-a [*n***]] [-e** *string* **] [-j[***n***]** *keyf***] [-o** *n.m ...***] [-t***c***]** *file1 file2*

join processes the text files *file1* and *file2*, each of which contains a relational data base. If either file name is '-', the standard input is used for that file.

For the purposes of **join**, a data base file contains a set of records, one per input line. Each record contains a number of *fields*. One field is differentiated as *key* field for each file. Each file must be sorted by key field, for example with **sort**.

By default, the key field is the first field in each record. The **-j** option changes the key field number to *keyf* for the desired file. In this and other options below, the optional file number *n* must be **1** to indicate *file1* or **2** to indicate *file2*. If no *n* is given, both *file1* and *file2* are assumed.

Normally, fields are separated by any amount of white space (blanks or tabs). Leading blanks or tabs are not considered part of the fields. With the **-t** option, the separator character is *c*. With this option zero-length fields are possible; every occurrence of the separator ends the previous field and starts a new one.

Output consists only of records for which the key field occurs in both files. As a consequence of the sorted order of the input, the output is also sorted by the key field. Each output record has first the key field, then each field from the *file1* record but the key field, and then each field from the *file2* record but the key field. Fields are separated in the output with the specified field character, or with a space character if no **-t** option was given. Output records are always terminated with a newline. Under the **-e** option, *string* is printed for each empty field.

The **-a** option enables printing of records found in only file *n*. If *n* is missing, unpaired records are printed from both input files. To output only certain fields, the **-o** option precedes a list of desired fields to print. Each element is of the form *n.m* where *n* is the file number and *m* is the field number.

For example,

```
	join -t: -j1 3 -o 1.3 2.4 1.4 1.1 2.2 filea fileb
```

joins **filea** and **fileb** which have fields separated by the colon (':') character. The join field number is 3 for **filea** and 1 (by default) for **fileb.** The selected five fields are produced in the output.

See Also

awk, comm, commands, sort, uniq

jrand48() — Random-Number Function (libc)

Return a 48-bit pseudo-random number as a long integer
long jrand48(*xsubi***)**
unsigned short *xsubi***[3];**

Function **jrand48()** generates a 48-bit pseudo-random number, and returns its upper 32 bits in the form of a **long**. The value returned is (or should be) uniformly distributed throughout the range of -2^31 through 2^31. *xsubi* is an array of three unsigned short integers from which the pseudo-random number is built.

See Also

libc, srand48()