NAME

monitor – prepare process execution profile

SYNOPSIS

#include <mon.h>

void monitor(int (∗*lowpc***)( ), int (**∗*highpc***)( ), WORD** ∗*buffer***, size_t** *bufsize***,**
size_t *nfunc***);**

MT-LEVEL

Safe

DESCRIPTION

monitor( ) is an interface to **profil( )**, and is called automatically with default parameters by any program created by **cc**(1B) −**p.** Except to establish further control over profiling activity, it is not necessary to explicitly call **monitor( )**.

When used, **monitor( )** is called at least at the beginning and the end of a program. The first call to **monitor( )** initiates the recording of two different kinds of execution-profile information: execution-time distribution and function call count. Execution-time distribution data is generated by **profil( )** and the function call counts are generated by code supplied to the object file (or files) by **cc**(1B) −**p.** Both types of information are collected as a program executes. The last call to **monitor( )** writes this collected data to the output file **mon.out**.

The name of the file written by **monitor( )** is controlled by the environment variable **PROFDIR**. If **PROFDIR** does not exist, the file **mon.out** is created in the current directory. If **PROFDIR** exists but has no value, **monitor( )** does no profiling and creates no output file. If **PROFDIR** is *dirname*, and **monitor( )** is called automatically by compilation with **cc** −**p**, the file created is *dirname***/***pid.progname* where *progname* is the name of the program.

lowpc and *highpc* are the beginning and ending addresses of the region to be profiled.

buffer is the address of a user-supplied array of **WORD** (**WORD** is defined in the header **<mon.h>**). *buffer* is used by **monitor( )** to store the histogram generated by **profil( )** and the call counts.

bufsize identifies the number of array elements in *buffer*.

nfunc is the number of call count cells that have been reserved in *buffer*. Additional call count cells will be allocated automatically as they are needed.

bufsize should be computed using the following formula:

```
size_of_buffer =
        sizeof(struct hdr) +
        nfunc * sizeof(struct cnt) +
        ((highpc-lowpc)/BARSIZE) * sizeof(WORD) +
        sizeof(WORD) − 1 ;
bufsize = (size_of_buffer / sizeof(WORD)) ;
```

where:

lowpc, *highpc*, *nfunc* are the same as the arguments to **monitor( )**;

BARSIZE is the number of program bytes that correspond to each histogram bar, or cell, of the **profil( )** buffer;

the **hdr** and **cnt** structures and the type **WORD** are defined in the header **<mon.h>**.

The default call to **monitor( )** is shown below:

monitor (&eprol, &etext, wbuf, wbufsz, 600);

where:

eprol is the beginning of the user's program when linked with **cc** −**p** (see **end**(3C));

etext is the end of the user's program (see **end**(3C));

wbuf is an array of **WORD** with *wbufsz* elements;

wbufsz is computed using the *bufsize* formula shown above with *BARSIZE* of 8;

600 is the number of call count cells that have been reserved in *buffer*.

These parameter settings establish the computation of an execution-time distribution histogram that uses **profil( )** for the entire program, initially reserves room for 600 call count cells in *buffer*, and provides for enough histogram cells to generate significant distribution-measurement results. For more information on the effects of *bufsize* on execution-distribution measurements, see **profil**(2).

EXAMPLES

To stop execution monitoring and write the results to a file, use the following:

monitor((int (*)( ))0, (int (*)( ))0, (WORD *)0, 0, 0);

Use **prof** to examine the results.

FILES

mon.out

SEE ALSO

cc(1B), **profil**(2), **end**(3C), **prof**(5)

NOTE

Additional calls to **monitor( )** after **main( )** has been called and before **exit( )** has been called will add to the function-call count capacity, but such calls will also replace and restart the **profil( )** histogram computation.