NAME

zprof, zcounts - transform ANSI-C programs to enable basic block profiling.

SYNOPSIS

zprof
[**--cond**|**-c**]
[**--help**|**-h**]
[**--prefix**=*PREFIX*|**-p***PREFIX*]
[**--preprocess**|**-P**]
[**--silent**|**-s**]
[**--version**|**-v**]
[**--typedef**=*ID*|**-t***ID*]
[*file*.c...]

zcounts

DESCRIPTION

zprof is a preprocessor which transforms a ANSI-C program to enable basic block profiling. When the transformed program is compiled and executed, it records the number of times each of its basic blocks is executed.

Each source file *file*.**c** is transformed to *file*_**BB.c** (to change the output name, see the **--prefix** option). The transformed program needs to be linked with the **zprof** library **libzprof.a**. When the transformed program is executed, it appends the execution counts of its basic blocks to the file **zprof.out**. The **zcounts** script uses the generated **zprof.out** file to produce an annotated version of the original files (with names of the form *file*.**c.bb**) with each line preceeded by its execution count. The **zprof.out** file can also be analyzed using standard utilities (see the **EXAMPLES** section).

If no source files are specified, **zprof** simply prints a summary of its options and exits.

OPTIONS

Zprof supports both traditional single-letter options and mnemonic long option names. Long option names are indicated using -- instead of -. Abbreviations for long option names are allowed as long as they are unique. When a long option like **--prefix** takes an argument, connect the option name and the argument with = or put the argument in the next word. Similarly, when a short option like **-p** takes an argument, put the argument value immediately after the option in the same word, or put the argument in the next word. Boolean argument values may be specified as **0** or **1**, **yes** or **no**, **on** or **off**.

--cond
-c Profile subexpressions of all ?: conditional expressions (default: **off**).

--help
-h Print summary of options and exit.

--prefix=*PREFIX*
-p*PREFIX*

Use *PREFIX* as prefix of all generated names in instrumented file and for forming the name of the output files (default: **_BB**).

--preprocess
-P Run the preprocessor (given by environmental var **CPP**) on the input file before **zprof** sees it (default: **on**).

--silent
-s Do not generate error messages (default: **off**).

--version
-v Print version information and exit.

--typedef=*ID*
-t*ID* Declare identifier *ID* to be a typedef in the global scope.

ENVIRONMENT

If preprocessing is not turned off using **--preprocess**=**no**, then the command used to run the C-preprocessor is taken from the environmental variable **CPP**. If **CPP** is not defined, then the name of a C compiler is sought in environmental variable **CC**, and each source file *file*.**c** is preprocessed using the command:

$CC -E *file*.**c**

If **CC** is also not defined, then the C-preprocessor is run using

cc -E *file*.**c**

EXAMPLES

Consider the simple prime number program from file **primes.c** shown below:

```
static int
prime(int n)
{
  int i;
  for (i= 2; i < n; i++) {
    if (n % i == 0) {
      return 0;
    }
  }
  return 1;
}

int
main()
{
  const int n= 1000;
  int i;
  for (i= 2; i <= n; i++) {
    if (prime(i)) {
      printf("%d0, i);
    }
  }
  return 0;
}
```

This can be profiled as follows:

```
$ zprof primes.c
$ cc primes_BB.c -lzprof -o primes_bb
$ rm -f zprof.out
$ ./primes_bb >/dev/null
$ zcounts
```

The first line transforms **primes.c** into **primes_BB.c** which has additional code for accumulating profile counts. The second line compiles the transformed file and links the object file with the **zprof** library to get an executable **primes_bb**. The third line ensures that there is no **zprof.out** file present in the current directory, as the execution counts from the next step are appended to it. The fourth line runs **primes_bb**, producing a **zprof.out** file

containing the execution statistics. The final line uses the statistics to produce an annotated version **primes.c.bb** shown below:

```
            static int
            prime(int n)
            {
              int i;
    999       for (i= 2; i < n; i++) {
    78022       if (n % i == 0) {
    831           return 0;
                }
              }
    168       return 1;
            }

            int
            main()
            {
              const int n= 1000;
              int i;
    1         for (i= 2; i <= n; i++) {
    999         if (prime(i)) {
    168           printf("%d0, i);
                }
              }
    1         return 0;
            }
```

In order to correctly parse C, it is necessary to recognize **typedef**s. Since **typedef**s are often defined in header files, it is necessary to parse all included header files --- this is typically done by running the C-preprocessor first. Unfortunately, on some systems like Linux, the system header files use non-ANSI extensions, which causes problems for **zprof** which expects ANSI-C. It is still possible to use **zprof** on such systems by using the C-preprocessor to remove the non-ANSI extensions. For example, the following works for profiling some programs under Linux.

```
$ CPP="gcc -E -D'__attribute(n)__=' -D__const=const" \
  zprof ...
```

The options **--preprocess=0**, **--silent** and **--typedef** may also be useful in such situations where a header which cannot otherwise be parsed defines only a few **typedef**s.

FILES

The format of the generated **zprof.out** file consists of lines of the form:

filename:*linenum*: *count*

specifying that line number *linenum* in file *filename* was executed *count* times. If there are multiple basic blocks associated with a particular source line, then multiple lines will be produced for the same *filename* and *linenum*.

The above format is amenable to easy processing by several tools:

+ The file can easily be sorted in descending order of counts:

  sort -t: -nr -2 zprof.out

+ Emacs compilation-mode can be used to interactively browse this file. **M-x compile** can be used to start compilation and the compilation command can be specified simply as **cat zprof.out**. Then the **M-`** command can be used repeatedly to position the cursor at each source line with its execution count shown in the compilation buffer.

AUTHOR

Copyright (C) 1997 Zerksis D. Umrigar

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

SEE ALSO

prof(1) **monitor**(3)