KPROF Free License Key Download [Latest]

KPROF Crack For PC

KPROF is a command-line tool that allows you to efficiently sample and profile programs using the Windows API, all with one command line. Sample programs are run with a specified sampling interval, and the result of sampling is stored in a heap allocated memory region. For more detailed profiling, call the KPROF API and profile an existing program, or record information about the program with stack traces and call stacks. KPROF is written in C++, and was originally written by Greg Raab. The original source is on his site: www.raabjournals.com/kprprof/ KPROF is a light weight program. It has been optimized for a small footprint, and does not depend on other library components or tools. It works with Windows 2000 and later. Sample / profile programs have a maximum of 10MB of code and data in memory at a time. This includes the heap space used by the sampler / profiler. KPROF includes: Sample program selection, including DLL injection Run / test functions and arguments Test events, threads, and synchronization primitives Memory profiling Call Graph generation Memory Call Graph generation Breakpoint generation and deletion Recording stack traces and call stacks Locking and unlocking samples C/C++ sample program injection Type enumerations Virtualization support HTTP

support for displaying HTML pages Misc. features KPROF also has several GUI based utilities. These are very basic, but do allow you to review samples using: A text editor KernProf KernProf KernProf: A GUI for KPROF Sample Programs The KPROF sample programs are listed at: www.raabjournals.com/kprprof/ If you select the normal Sample without Append directory at the start of the download, you get the following samples: KPROF_Timing.exe: A minimal timer that runs on a Windows system. This should be of help if you are in the habit of time measurement in your code. KPROF_HeapDump.exe: A function call / heap dump tool. This is designed to quickly dump all memory in a program into a file. KPROF_Help.exe: A program to browse the KPROF help files. KPROF_Exchange.exe: A simple, easy to use C/C++/VB/ASP exchange server program. The sample program uses

KPROF Free Registration Code Free

- - is newline - - \z is to escape a character - - \p e.g. is function name - - \r is to escape a character - - > after command is executed, this is how the output of the command is displayed - - \c concatenate the command

line into a single string. - - \t screen breaks - - \d is to dump a list of function names - - \e displays the current line number and marks current position - - | has no meaning - - c...Continue to the next line, once it is entered with a | - - :c...Continue to the next line, but don't advance the cursor to it - - :n...Begin a new line, not just a new tab - - # comment, remove this line - - :w...write - - :q...quit What does the last line do? What is the difference between *c* and *c*:? In the first *c* the output of the command is concatenated to the line you are currently on. In the second *c* the output of the command is displayed in the window. The *c* is used for debugging, it is useful when you want to look at the output of the command, without having to switch to that window. The difference between the first and second *c* is whether the output of the command is displayed in the window or is shown as a concatenation to the current line. If you want to go to the next line use: To advance to the next line, use: If you want to go to a line, where there is a specific character, then: To advance to the next line, use: If you want to go to the next line, and remove the character you are currently on, then: To advance to the next line, use: If you want to go to the next line, advance the cursor to it, but don't enter it, then: To advance to the next line, use: If you want to advance to the next line, advance the cursor to it, and enter it,

then: To advance to the next line, use: If you want to advance to the next line, and enter it, and remove the character you are currently on, then: 2edc1e01e8

KPROF Crack + Free

KPROF is a convenient way to profile and analyze performance problems. Its functionality and design closely parallels the traditional Win32 method of instrumenting with performance counters. In addition to traditional execution sampling, it can also collect branch profiling information. Differences from Windows: It is free. Windows profilers are generally proprietary and cost thousands of dollars. KPROF's core functionality is tightly integrated with KDevelop, and is available in the menus, along with a comprehensive set of debugging tools. It is therefore easy to launch profiling sessions from within the IDE. Features: KPROF can be launched from within the IDE, and operates in the background. It uses the windows task manager to collect performance information, and it can also measure timing information using the Mwait functions in the msvcrt.dll. Runtime Profiling KPROF can profile running applications. As the profiling session progresses, hotspots are identified by KPROF's HotSpot reporter. When a hotspot is identified, information about the profile events is logged into the console and displayed. In addition, if you have allocated KPROF as the application debugger for your program, KPROF will also show the disassembled code

at the hotspot location. The disassembled code allows you to easily see the assembly language and examine the call stack. HotSpot Resolution KPROF lets you analyze the assembly language directly. If there is a problem in the code that you are analyzing, you can launch KPROF on the code while it is executing. The HotSpot reporter will use KPROF's runtime profiling data to identify the relevant hotspots in the code and will show you their associated assembly language and function call stack. Branch Profiling KPROF's Branch Reporter can be used to collect information about conditional branch instructions in the code. KPROF's Branch Reporter can examine the target addresses of conditional branch instructions in the code and can also compare their relative distances from other instructions. You can also use KPROF to identify which conditional branches in the code are the most important. Profile Distribution KPROF's HotSpot reporter can graphically show how hotspots are distributed across the code. If you choose, you can also view the graph by listing all the hotspots in the code. Custom Profiling KPROF can be customized to suit your specific profiling needs. It is possible to profile any code using any of KPROF's reporters. It is possible to collect profiling information for any address in the code

What's New in the?

KPROF, or kurtosis-based profiler, was written by Eric Braun (kurtosis at mtnsa.usc.edu). If you have an executable on disk, it will execute that (the third command line option). Otherwise, you can get the tool from the Internet at kurtosis.sourceforge.net. It is also availible for Win32, OS/2, BeOS, Amiga, and Mac. Vista Information: Kurtosis-based profiling uses the kurtosis value of a program's execution time to identify where it spends its time. In most cases, it's the difference between the 1st and 2nd moments. In non-normal distributions, the mean is a fairly coarse measure of where most of the weight is. Kurtosis provides a much finer measure. Thus, the typical application of kurtosis-based profiling is to find where the heaviest work is happening. The output is a quick

report of all the hot spots in the code. How to Use: You run KPROF once for each executable on disk, and then exit. The program will run for a few seconds before beginning to give you results. In the initial stages, it will give you lots of output. KPROF Output: In the initial stages of execution, KPROF will give a listing of all the hot spots in the program. The list will be a single line with a bunch of numbers in the text. If the program is small, it will list all the hot spots. If the program is large, KPROF will first calculate a rough estimate of the hot spots using information from the lowest-kurtosis hot spot. This estimate will be more accurate and a better guide than just listing the hot spots (which would take longer to do). After the rough estimate, KPROF will show you all the hot spots, along with a graphical view of the program's graph of execution time versus execution time kurtosis. KPROF will also show you the names of each hot spot and in which file (line) they are found. PID/Unique Identifier: If you have a PID for the executable that you are profiling, you can use that to latch onto it and determine where most of the work is going to be done (i.e., in which file (line) on disk). If you don't know the PID, you can find it in Task Manager. If you have more than one PID, KPROF will average the results of all the different processes in Task Manager. You can then latch onto this PID using the 2nd command line

option. KPROF Options: -d - Debugging information. Shows some of the assembly (if it was compiled with debug information). -h - Show

System Requirements:

OS: Windows XP, Windows Vista, Windows 7, Windows 8 CPU: Intel Pentium 4 1.7 GHz or greater RAM: 2 GB CD-ROM: DirectX 9.0c compatible and CD-ROM drive HDD: 300 MB free space Sound: Microsoft Sound System compatible sound card Video Card: DirectX 9.0c compatible video card with 2048x2048 resolution, minimum display drivers V 1.9.0 If you are running Windows Vista or newer you will

Related links: