NAME

blinkenlights — x86-64-linux visualizer

SYNOPSIS

blinkenlights [*options*] *program* [*argv1...*]
blinkenlights [*options*] **-0** *program* [*argv0...*]

DESCRIPTION

blinkenlights is a terminal user interface for debugging x86_64-linux (or i8086) programs on POSIX platforms. It focuses on visualizing how program execution impacts memory. It uses UNICODE IBM Code Page 437 characters to display binary memory panels, which change as you step through your program's assembly code. These memory panels may be scrolled and zoomed using your mouse wheel. **blinkenlights** also permits reverse debugging, where scroll wheeling over the assembly display allows the rewinding of execution history.

ARGUMENTS

The *program* argument is a `PATH` searched command, which may be:

- An x86_64-linux ELF executable (either static or dynamic)
- An Actually Portable Executable (either MZqFpD or jartsr magic)
- A flat executable (if *program* ends with .bin, .img, or .raw)
- A shell script whose #!interpreter meets the above criteria

The **-0** flag allows `argv[0]` to be specified on the command line. Under normal circumstances,

```
blinkenlights cmd arg1
```

is equivalent to

```
execve("cmd", {"cmd", "arg1"})
```

since that's how most programs are launched. However if you need the full power of execve() process spawning, you can say

```
blinkenlights -0 cmd arg0 arg1
```

which is equivalent to

```
execve("cmd", {"arg0", "arg1"})
```

OPTIONS

The following options are available:

-h Prints condensed help.

-v Shows **blinkenlights** version and build configuration details.

-N Enables natural scrolling.

-H Disables syntax highlighting.

-r Puts your virtual machine in real mode. This may be used to run 16-bit i8086 programs, such as SectorLISP. It's also used for booting programs from Blinkenlights's simulated BIOS.

-b *addr*
Pushes a breakpoint, which may be specified as a raw hexadecimal address, or a symbolic name that's defined by your ELF binary (or its associated `.dbg` file). When pressing `c` (continue) or `C` (continue harder) in the TUI, Blink will immediately stop upon reaching an instruction that's listed as a breakpoint, after which a modal dialog is displayed. The modal dialog may be cleared by `ENTER` after which the TUI resumes its normal state.

-w *addr*
: Pushes a watchpoint, which may be specified as a raw hexadecimal address, or a symbolic name that's defined by your ELF binary (or its associated `.dbg` file). When pressing `c` (continue) or `C` (continue harder) in the TUI, Blink will immediately stop upon reaching an instruction that either (a) has a ModR/M encoding that references an address that's listed as a watchpoint, or (b) manages to mutate the memory stored at a watchpoint address by some other means. When Blinkenlights is stopped at a watchpoint, a modal dialog will be displayed which may be cleared by pressing `ENTER` after which the TUI resumes its normal state.

-t
: May be used to disable Blinkenlights TUI mode. This makes the program behave similarly to the `blink`(1) command, however not as good. We're currently using this flag for unit testing real mode programs, which are encouraged to use the `SYSCALL` instruction to report their exit status.

-s
: Enables system call logging (repeatable).

This will emit to the log file the names of system calls each time a SYSCALL instruction in executed, along with its arguments and result. System calls are logged once they've completed, so that the result can be shown.

System calls are also sometimes logged upon entry too, in which case `syscall(arg) -> ...` will be logged to show that the system call has not yet completed. Whether or not this happens depends on how many times the **-s** flag is supplied.

- Passing **-s** (once) will only log the entries of system calls that are defined as having read + write parameters (e.g. poll, select)
- Passing **-ss** (twice) will also log the entries of cancellation points that are likely to block (e.g. read, accept, wait, pause).
- Passing **-sss** (thrice) will log the entries of system calls that POSIX defines as cancellation points but are unlikely to block (e.g. write, close, open) which we try to avoid doing in order to reduce log noise.
- Passing **-ssss** (4x) will log the entry of every single system call, even harmless ones that have no business being emitted to the log twice (e.g. sigaction). This should only be useful for the Blink dev team when the rare need arises to troubleshoot a system call that's crashing.

System call logging isn't available in `MODE=rel` and `MODE=tiny` builds, in which case this flag is ignored.

-j
: Enables Just-In-Time (JIT) compilation. This will make Blinkenlights go significantly faster, at the cost of taking away the ability to step through each instruction. The TUI will visualize JIT path formation in the assembly display; see the JIT Path Glyphs section below to learn more. Please note this flag has the opposite meaning as it does in the `blink`(1) command.

-m
: Enables the linear memory optimization. This makes **blinkenlights** capable of faster emulation, at the cost of losing some statistics. It no longer becomes possible to display which percentage of a memory map has been activated. Blinkenlights will also remove the commit / reserve / free page statistics from the status panel on the bottom right of the display. Please note this flag has the opposite meaning as it does in the `blink`(1) command.

-L *path*
: Specifies the log path. The default log path is *blink.log* in the current directory at startup. This log file won't be created until something is actually logged. If logging to a file isn't desired, then -L /dev/null may be used. See also the **-e** flag for logging to standard error.

-C *path*
: Launch *program* in a chroot'd environment. This flag is both equivalent to and overrides the `BLINK_OVERLAYS` environment variable.

−z [repeatable] May be specified to zoom the memory panels, so they display a larger amount of memory in a smaller space. By default, one terminal cell corresponds to a single byte of memory. When memory has been zoomed the magic kernel is used (similar to Lanczos) to decimate the number of bytes by half, for each **−z** that's specified. Normally this would be accomplished by using `CTRL+MOUSEWHEEL` where the mouse cursor is hovered over the panel that should be zoomed. However, many terminal emulators (especially on Windows), do not support this xterm feature and as such, this flag is provided as an alternative.

−Z Prints internal statistics to standard error on exit. Each line will display a monitoring metric. Most metrics will either be integer counters or floating point running averages. Most but not all integer counters are monotonic. In the interest of not negatively impacting Blink's performance, statistics are computed on a best effort basis which currently isn't guaranteed to be atomic in a multi-threaded environment. Statistics aren't available in MODE=rel and MODE=tiny builds, in which case this flag is ignored.

−V [repeatable] Increases verbosity.

−R Disables reactive error mode.

KEYBOARD SHORTCUTS

The following keystrokes are recognized by the user interface:

? Shows help dialog.

q Quit.

s Step. This executes a single instruction, stepping forward by one.

n Next. This is the same as **s** (Step) except it won't recurse into `CALL` instructions.

c Continue. This will step automatically and display an animation of the program execution as it progresses. In continue mode, Blinkenlights will execute as many instructions as possible, and only render a limited number of 60 frames per second to the terminal. Snapshots are captured in the background of the in-between steps that aren't displayed. They may still be viewed by pausing execution using **CTRL−C** (Interrupt) and then pressing UP arrow to scroll backwards through execution history. For large programs, this animation can be sped up (at the cost of losing frames) by using the **CTRL−T** (Turbo) shortcut.

C Continue Harder. This will execute the program in the background as quickly as possible until some kind of halting event occurs, such as exit_group() being called, or a segmentation fault. No animation is displayed during this time.

CTRL−C Interrupt. Pressing this key will interrupt the TUI animation when in **c** (Continue) mode. Control will then return to the main interface, and keyboard shortcuts such as **s** (Step) may once again be used. Pressing **CTRL−C** also has the same effect if the embedded teletypewriter is blocked on a read() call, awaiting keyboard input.

CTRL−T Turbo. The turbo key may be pressed multiple times to specify how many steps should happen per frame in **c** (Continue) mode. Each time this key is pressed, the status bar on the bottom left-hand side of the display will be updated with the current speed, which defaults to 1.

ALT−T Slowmo. The slowmo keyboard shortcut has the opposite effect of **CTRL−T** (Turbo) in the sense that it slows down the speed of the **c** (Continue) mode animation. Each time this key is pressed, the status bar on the bottom left-hand side of the display will be updated with the current speed, which defaults to 1, and descends into negative numbers. Positive values are defined as the number of steps per frame. Negative numbers will result in sleep delays being inserted between steps.

p Profiling Mode. Pressing the **p** key will cause the TUI to cycle between the profiling and backtrace panels. When **blinkenlights** is running, it maintains a naive count of the number of assembly opcodes executed at each memory address. When the profiling panel is displayed, those counters will be grouped by function, ranked, and displayed as a percentage of the total. This is intended to help identify, in real time as execution progresses, which functions are execution hotspots.

t SSE Type. When the SSE panel is being displayed, the TUI will determine the type of each XMM register based on the instructions used. The three different types defined for this purpose are (1) integral, (2) single, and (3) double.

T SSE Size. When the SSE panel is being displayed, pressing this key will cycle the vector element size of XMM registers in integral mode. The following sizes are defined: 1 (byte), 2 (word), 4 (dword), 8 (qword).

x SSE Radix. When the SSE panel is being displayed, pressing this key will cycle the display of XMM registers currently in integral mode, so that they're displayed as either (1) hexadecimal, (2) characters (CP437), or (3) decimal. Floating point XMM registers aren't impacted, unless the **t** (Sse Type) key is pressed beforehand to cycle them into integral mode.

M The **M** key may be pressed to toggle xterm mouse tracking. This may be useful for terminals that do not allow copying and pasting terminal content when mouse tracking is enabled (try shift+drag too).

MOUSEWHEEL
Scroll. Using the mouse wheel has a different effect depending on which panel the mouse cursor is currently hovering over. When the mouse is above the disassembly panel, scrolling the mouse wheel will rewind and replay the history of program execution. When above memory panels, mouse wheel will display different memory addresses.

CTRL−MOUSEWHEEL
Zoom. On platforms such as Linux that support the necessary xterm escape code for doing this, using mousewheel while holding down the control key when the mouse cursor is hovering above one of the memory panels, will cause that memory panel to become zoomed. Under normal circumstances, each TTY cell corresponds to a single byte of memory. Zooming by one notch will cause each cell to display two bytes of memory. Then four. Then eight. This is accomplished by successively applying an image scaling algorithm to the adjacent memory.

CTRL−Z Stop. Pressing this shortcut will place **blinkenlights** in the background and return control to the shell. You may resume your debugging session later by running the **fg** command.

CTRL−L Refresh. Pressing this keyboard shortcut will cause the display to be redrawn. This may be useful on the oft chance the terminal state becomes corrupted.

CTRL−D EOF (End Of File). This keystroke has two different meanings depending on context. When control is held by the debugger TUI, this will ask **blinkenlights** to exit. When control is held by the guest program invoking read() in the teletypewriter, this will close the standard input handle.

ENVIRONMENT

The following environment variables are recognized:

`BLINK_LOG_FILENAME`
may be specified to supply a log path to be used in cases where the **−L** *path* flag isn't specified. This value should be an absolute path. If logging to standard error is desired, use the **−e** flag.

`BLINK_OVERLAYS`
specifies one or more directories to use as the root filesystem. Similar to `PATH` this is a colon delimited list of pathnames. If relative paths are specified, they'll be resolved to an absolute path at

startup time. Overlays only apply to IO system calls that specify an absolute path. The empty string overlay means use the normal / root filesystem. The default value is `:o` which means if the absolute path `/$f` is opened, then first check if `/$f` exists, and if it doesn't, then check if `o/$f` exists, in which case open that instead. Blink uses this convention to open shared object tests. It favors the system version if it exists, but also downloads `ld-musl-x86_64.so.1` to `o/lib/ld-musl-x86_64.so.1` so the dynamic linker can transparently find it on platforms like Apple, that don't let users put files in the root folder. On the other hand, it's possible to say `BLINK_OVERLAYS=o:` so that `o/...` takes precedence over `/...` (noting again that empty string means root). If a single overlay is specified that isn't empty string, then it'll effectively act as a restricted chroot environment.

EXIT STATUS

The **blinkenlights** command passes along the exit code of the *program* which by convention is 0 on success or >0 on failure. In the event that **blinkenlights** fails to launch *program* the status 127 shall be returned.

SEE ALSO

`blink`(1)

AUTHORS

Justine Alexandra Roberts Tunney <jtunney@gmail.com>

QUIRKS

Blinkenlights TUI currently isn't suitable for debugging programs that spawn threads / processes. While such programs may be debugged, separate threads and processes can not be controlled reliably by the keyboard.