ltrace Library call tracer

CONCEPTS

ltrace shows calls into shared libraries (libc, libssl...) and prints their arguments. Use strace for kernel syscalls only.

Your program main binary
- **Shared libraries** ltrace: library calls
  - **Kernel** strace: syscalls (ltrace -S)

Only calls through the PLT are seen. Statically linked programs and calls inside one library are invisible unless you use -x.

Output goes to stderr, so redirect with 2> or use -o. Without -f a forking program is traced only in the parent.

RUN AND ATTACH

ltrace *<cmd>* **[args]**
Trace a new command

ltrace -p *<pid>*
Attach to a running process

ltrace -f *<cmd>*
Follow forks and clones

ltrace -p *<pid>* **-p** *<pid2>*
Attach to several processes

ltrace -u *<user>* *<cmd>*
Run the command as another user

ltrace -o *<file>* *<cmd>*
Write the trace to a file

ltrace -h / -V
Show help / version

FILTER CALLS

ltrace -e *<func>* *<cmd>*
Trace only this function

ltrace -e *<f1>*+*<f2>* *<cmd>*
Trace several functions

FILTER CALLS (cont.)

ltrace -e @libc.so* *<cmd>*
Only calls into libc

ltrace -e *<func>*@*<lib>* *<cmd>*
Function from a given library

ltrace -e '/^str' *<cmd>*
Regex on function names

ltrace -e -*<func>* *<cmd>*
Trace everything except this

ltrace -l *<lib>* *<cmd>*
Calls into libraries matching a pattern

ltrace -e @MAIN *<cmd>*
Calls from the main binary only (default)

ltrace -L *<cmd>*
Trace no library calls (with -S: syscalls only)

ltrace -x '*<pattern>***'** *<cmd>*
Trace symbol-table functions, internal calls too

ltrace -S *<cmd>*
Show system calls as well

OUTPUT FORMAT

ltrace -s *<n>* *<cmd>*
Longest printed string (default 32)

ltrace -C *<cmd>*
Demangle C++ names

ltrace -i *<cmd>*
Print the instruction pointer

ltrace -n *<n>* *<cmd>*
Indent nested calls by n spaces

ltrace -A *<n>* *<cmd>*
Cap printed array elements

ltrace -d *<n>* *<cmd>*
Limit struct pointer nesting depth

ltrace -a *<col>* *<cmd>*
Align return values at a column

ltrace -b *<cmd>*
Hide received signals

OUTPUT FORMAT (cont.)

ltrace -w *<n>* *<cmd>*
Backtrace of n frames per call

ltrace -F *<path>* *<cmd>*
Extra prototype config files or dirs

TIMING AND STATISTICS

ltrace -c *<cmd>*
Count calls and time per function

ltrace -T *<cmd>*
Time spent inside each call

ltrace -r *<cmd>*
Relative timestamp per line

ltrace -t *<cmd>*
Time of day per line

ltrace -tt *<cmd>*
Time of day with microseconds

ltrace -ttt *<cmd>*
Epoch seconds with microseconds

ltrace -c -S *<cmd>*
Summary including system calls

RECIPES

Where does a program allocate?

```
ltrace -e
malloc+calloc+realloc+free -o
mem.log <cmd>
grep -c malloc mem.log
# unmatched malloc/free pairs hint
at a leak
```

Which strings does it compare?

```
ltrace -e
strcmp+strncmp+strcasecmp -s 100
<cmd> 2>&1 | head
# often reveals hard-coded
passwords or option names
```

Profile library use of a whole tree

```
ltrace -f -c -o summary.txt <cmd>
sort -k2 -n -r summary.txt | head
```

Library and kernel side by side

```
ltrace -f -S -tt -o both.log <cmd>
# library calls and syscalls
interleaved with timestamps
```