

Buildsome

The first true build system?

eyal.lotem@elastifile.com

Why not make?

- 2-phased build
 - Generate entire dependency graph from specification
 - Iterate graph to build required targets
- This is not enough!
- Auto-generated code + automatic dependencies means we need an arbitrary number of phases

Why not ninja/waf/shake/make?

- Ninja, waf, shake, make all require a reliable full specification of dependencies
- Such a specification is notoriously hard to specify correctly
- Example problem: Python imports
- Example problem: Include scan with multiple directories
- Cannot specify dependency on the **inexistence** of files
- Cannot maintain a global cache of previous builds

Why not tup?

- tup verifies the input/output specification for **local** input/output files to help with correctness
 - Still requires maintaining the full spec
 - Does not track global dependencies
- tup lets you over-specify dependencies
 - This unnecessarily sacrifices parallelism
- Does not support building specified sub-trees
- Cannot specify dependency on the **inexistence** of files
- Cannot maintain a global cache of previous builds

Why Buildsome?

- Require a full specification of **outputs only**
 - Even these are verified for correctness
- Optional, partial specification of inputs
 - Can be used for some extra efficiency in some cases
- Auto-detect inputs for both convenience and safety
 - Easy to specify Makefiles
 - No need to write elaborate, complicated dependency scanners!
 - No more frustrating “make clean” or cryptic build bugs!
- Even global/system inputs are detected.
 - Upgrade compiler will rebuild relevant parts
- Cares about correctness! Will detect modification of a source file during a build

Limitations

- Adds a small runtime overhead penalty to detect the inputs (~10%)

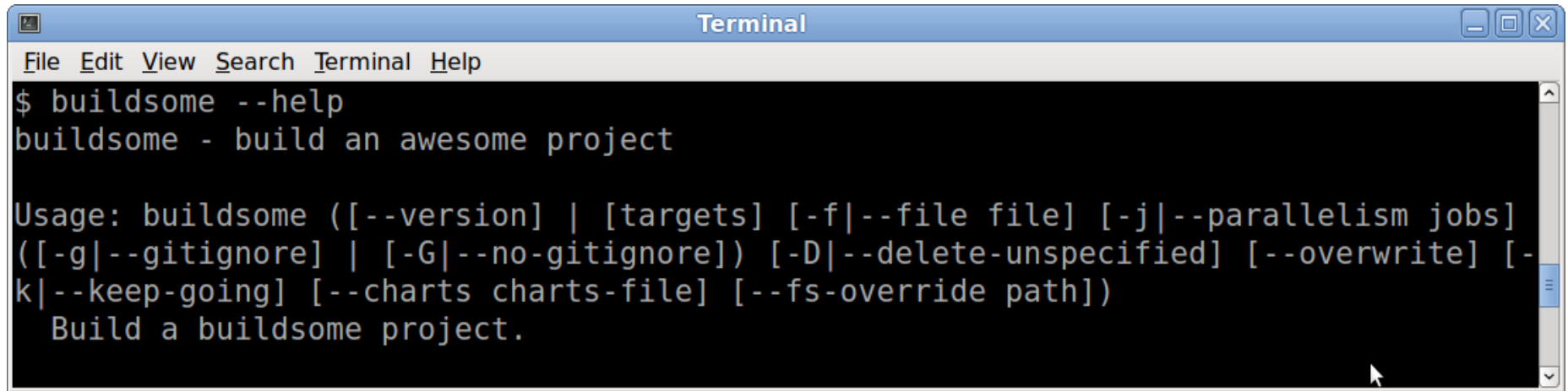
Imperfections

- Current implementation uses LD_PRELOAD hooks (not fully reliable)
- Can only detect **most** file system dependencies. Various **stat** fields cannot be considered inputs (e.g: mtime) but may theoretically be
- Cannot detect non-file-system inputs (e.g: socket interfaces)

Makefile

- Makefile-like format
 - Supports target syntax, including order-only inputs
 - Supports Make's metavariables
- More strict than Makefile about variables (variable name typos are errors!)
- New “local {” construct to avoid leaking unwanted changes to variables
- Miscellaneous other differences
- Future versions may abandon Make format for more powerful specifications

Features



```
Terminal
File Edit View Search Terminal Help
$ buildsome --help
buildsome - build an awesome project

Usage: buildsome ( [--version] | [targets] [-f|--file file] [-j|--parallelism jobs]
[[-g|--gitignore] | [-G|--no-gitignore]] [-D|--delete-unspecified] [--overwrite] [-k|--keep-going]
[--charts charts-file] [--fs-override path])
    Build a buildsome project.
```

- Parallel support
- Maintains .gitignore
- Can create a chart of build times
- Default target is “CWD/default”, not first in Makefile
 - Can simply run “buildsome” in your own development or test directory
- Replay old commands outputs & warnings
 - No need for -Werror!
- Outputs parsable line number information for IDE integration

Demo!