not sucking in the unix environment

unix weapons school

SCRIPTING

Bash grammar[a]

[a] This does not conform to `bash(1)` as of 4.2; I believe the official grammar to be inaccurate.

compound::= (<list>) | {<list>;} | ((<arithexpr>)) | [[<condexpr>]] |
`for` <name> [`in` <word> ... ;] `do` <list> ; `done` |
`for` ((<expr> ; <expr> ; <expr>)) ; `do` <list> ; `done` |
`select` <name> [`in` <word>] ; `do` <list> ; `done` |
`case` <word> `in` [[(<pattern> [| <pattern>...])] <list> ;;] ... `esac` |
`if` <list> ; `then` <list> ; [`elif` <list> ; `then` <list> ;] ... [`else` <list> ;] `fi` |
`while` <list> **FIXME** |
`until` <list> **FIXME**

function::= <name> () <compound> [<redirection>] |
`function` name [()] <compound> [<redirection>]

coproc::= `coproc` <name>

list::= <pipeline> [; | & | && | || <pipeline> ...] [; | &]

pipeline::= [`time` [-p]] [!] <command> [| | |& <command> ...]

arithexpr::=

condexpr::=

command::=

Shell scripts

- POSIX conformance: probably not worth it
- First line: `#!/bin/sh` or `/usr/bin/env` *shell*
- Second line: `set -e || exit 1`
- Pipelines only fail if the last command fails[1]
- Catch subshell errors via assignment
- `trap` on signals and pseudo-signal 0[2]

[1] Unless `-o pipefail` (bash-only) is used.

[2] bash allows `trap`ping EXIT as a synonym for 0.

Redirection

- Pipe creation occurs prior to redirection
- Redirect to stderr: `>&2`
- Functions can be redirected at definition, *and* at invocation
- Pipes can't target functions (use `read` if you must)
- Redirect both stdout and stderr: `>` *redir* `2>&1`
- Send stderr to stdin through pipe: `|&` (`2>&1 |`)
- Duplicate to stdout and a file: `| tee` *file*
- Append: `>>`*file*
- Dup and append: `| tee -a` *file*
- `/dev/std{in, out, err}`: stdio in a filename context

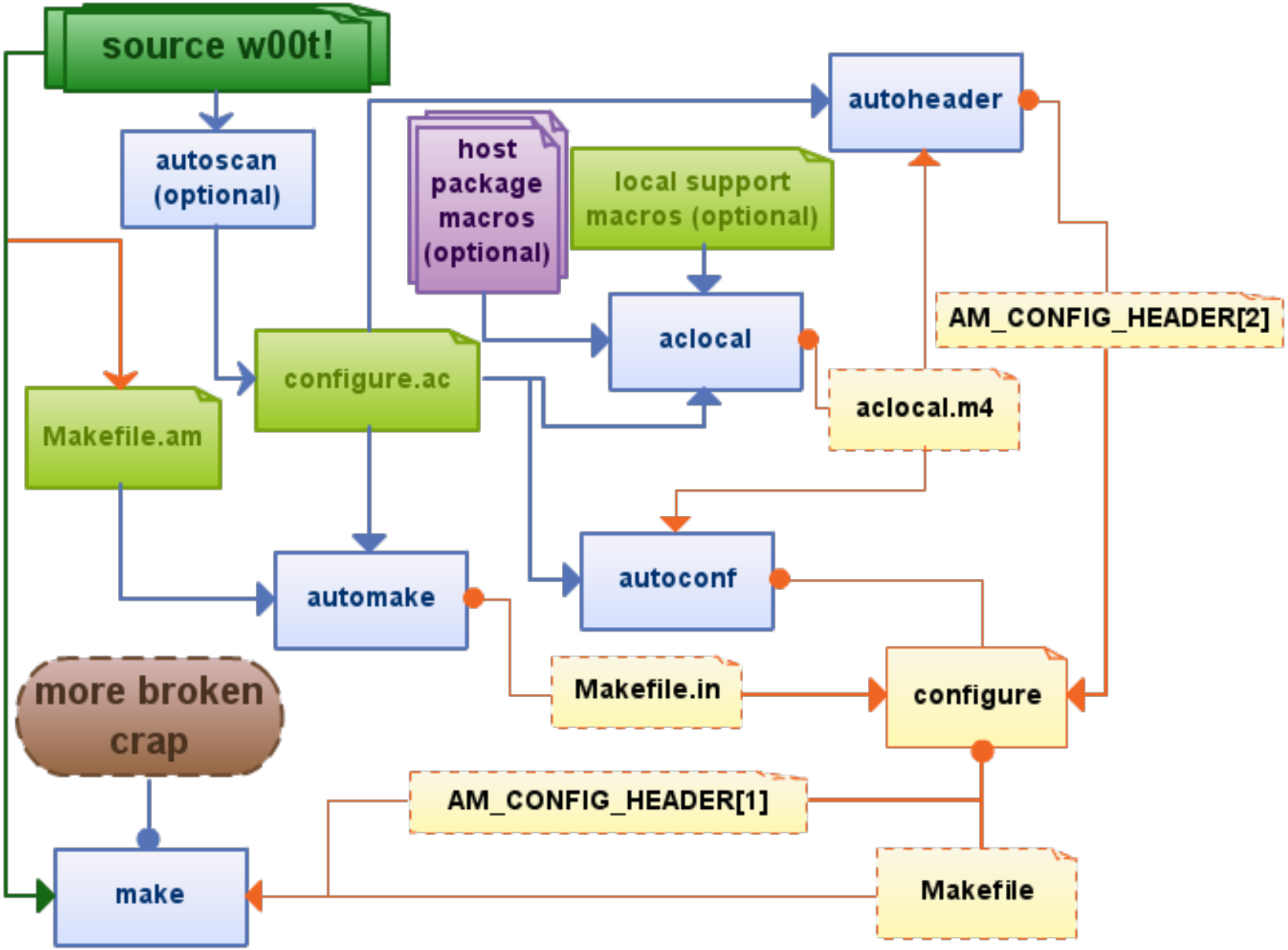
Perl

Please don't write perl.

FIXME approximately 5 more pages...

C/C++

GNU Autotools in one terrible diagram



FIXME approximately 10 more pages...

Recommended reading

- Dennis Ritchie. “Evolution of the Unix Time-sharing System” (1979).
- Neil Brown. “Ghosts of UNIX Past” LWN, in four parts (2004).
- Tom Christiansen. “`csh` Programming Considered Harmful” (1995).
- Mendel Cooper. “Advanced Bash-Scripting Guide” (2012).
- Peter Krumins. “`sed` One-Liners Explained” (2008).
- Peter Krumins. “`awk` One-Liners Explained” (2008).
- Steve Yegge. “Ancient Languages: Perl” (2004-12-04).
- “Implementing a Job Control Shell” GNU libc Info pages.
- Neal Stephenson. *In the Beginning was the Command Line* (1999).

Hack on!

"One of the main causes of the fall of the Roman Empire was that, lacking zero, they had no way to indicate successful termination of their C programs." —Robert Firth

"It's easier to port a shell than a shell script." —Larry Wall