Ninja で作る自作（？）ビルドシステム

自己紹介

- 名前:とやま ようた
- GitHub, Twitter: raviqqe
- 宗教: Vim

ビルドシステムとは

- ソースコードから効率よくバイナリを生成するためのソフトウェア
- 最近のトレンド：Task-based か Artifact-based へ
  - Make -> Bazel
- 言語ごとに異なる
  - C: Make, CMake, ...
  - Rust: Cargo
  - Go: `go build`

前に作ったビルドシステム

- Ein では手書きだった
  i. 全てのモジュールをスキャン
  ii. 依存関係グラフを生成
    - 相互再帰チェック
  iii. ビルドターゲットのソート
  iv. モジュールを順番にビルド
- 並列ビルドとかはもうちょっと面倒
- 新しい言語 Pen を作った
  - 外部のビルドシステムを利用したい

Ninja

- https://ninja-build.org/
- シンプルな Make
  - "Where other build systems are high-level languages Ninja aims to be an assembler."
- Make に比べて速い
  - Chrome や LLVM に使われているらしい
  - CMake のバックエンドのひとつ

Ninja (続)

- 元々 C や C++ のプロジェクトのために作られた
  - モジュールの実装 (.c ファイル) とインターフェース (.h ファイル)が別の言語
  - モジュール間のアーティファクト依存がない
- 最近の言語だと実装とインターフェースが同じファイルに書かれる
  - モジュール間のアーティファクト依存がある
  - Dynamic dependencies という機能が実装された
  - ビルド結果によって依存関係をその場で変えられる

Ninja を使ったビルドシステム

1. 全てのパッケージをダウンロード
    - 最初の一回のみ
2. 全てのパッケージ内の全てのモジュールをスキャン
    - 外部パッケージに対しては最初の一回のみ
3. それらをビルドする Ninja スクリプトを生成
4. Ninja を走らせる

.ninja スクリプトの例

```
ninja_required_version = 1.10
builddir = .pen
rule compile
  command = pen compile $in $out
  description = compiling module of $source_file
rule llc
  command = /home/raviqqe/.homebrew/bin/llc -O3 -tailcallopt -filetype obj -o $out $in
  description = generating object file for $source_file
rule resolve_dependency
  command = pen resolve-dependency -p $package_directory $in $object_file $out
  description = resolving dependency of $in
build .pen/objects/df9ec.bc .pen/objects/df9ec.json: compile Bar.pen .pen/objects/df9ec.dep || .pen/objects/df9ec.dd
  dyndep = .pen/objects/df9ec.dd
  source_file = Bar.pen
build .pen/objects/df9ec.o: llc .pen/objects/df9ec.bc
  source_file = Bar.pen
build .pen/objects/df9ec.dep .pen/objects/df9ec.dd.dummy: resolve_dependency Bar.pen
  package_directory =
  object_file = .pen/objects/df9ec.bc
build .pen/objects/df9ec.dep.dummy .pen/objects/df9ec.dd: resolve_dependency Bar.pen
  package_directory =
  object_file = .pen/objects/df9ec.bc
...
default .pen/objects/df9ec.o .pen/objects/91229.o
```

依存関係の解決

- 生成されたスクリプトにはモジュール間の依存関係は書かれていない
- 各モジュールに対して .dd ファイルをコンパイルし動的に依存関係を Ninja に伝える
  - 各モジュールのオジェクトファイルが依存するモジュールのインターフェースファイルのリスト

.dd ファイルの例

```
ninja_dyndep_version = 1
build .pen/objects/df9ec1801702a48a.bc: dyndep | .pen/objects/9122981f18913b17.json
```

実装

- `ninja_build_script_compiler.rs`
- `ninja_build_script_dependency_compiler.rs`

まとめ

- Ninja を使うと自作言語用のビルドシステムが簡単に作れる
- しかも速い

おまけ

- バグを踏んだ
- https://github.com/ninja-build/ninja/issues/1988