Rua 使用手册

bzhao

2025-02-13

目录

1 mkinfo

`rua mkinfo` 用于生成目标平台的构建指令。该指令包含了众多常用的 make 变量，用于方便地定制构建行为。例如：通过传入 `-6/--ipv6` 选项，令构建目标中包含 `ipv6` 字样；通过传入 `-w/--webui` 选项，将在指令中的 `NOTBUILDWEBUI` 变量设置为 `0`（带 WebUI）。

1.1 用法

用户可通过 `rua mkinfo -h` 查看帮助信息：

```
Get all matched makeinfos for product

Usage: rua mkinfo [OPTIONS] <PRODUCT>

Arguments:
  <PRODUCT>  Product name, such as 'A1000'. Regex is also supported, e.g. 'X\d+80'

Options:
  -4, --ipv4                       Build with only IPv4 enabled
  -6, --ipv6                       Build with IPv6 enabled
  -g, --coverage                   Run coverage
  -c, --coverity                   Run coverity
  -d, --debug                      Build in debug mode (default is release mode)
      --format <FORMAT>            Output format for makeinfos [default: list] [possible values: csv, json, list, tsv]
  -p, --password                   Build with shell password enabled
  -w, --webui                      Build with WebUI enabled
  -s, --image-server <IMAGE-SERVER>  Server to upload the output image to [possible values: b, s]
  -h, --help                       Print help

Examples:
  rua mkinfo A1000      # Makeinfo for A1000 without extra features
  rua mkinfo -6 A1000   # Makeinfo for A1000 with IPv6 enabled
  rua mkinfo -6w 'X\d+' # Makeinfos for X-series products with IPv6 and WebUI enabled using regex pattern
```

1.2 示例

1. `rua mkinfo -6 A1000`：生成 A1000 平台的构建指令，启用 IPv6 支持

```
bzhao in build18-bj in ~/repos/REL_R11
> rua mkinfo -6 A1000
1 matched info:
================================================================================================================================================================================
Product   : SG-6000-A1000
Model     : HS_TYPE_SG6000_A1000
Family    : HS_PRODUCT_FAMILY_FIREWALL
Platform  : HS_A1000
Target    : a-dnv-ipv6
Directory : products/ngfw_as/
Command   : hsdocker7 "make -C products/ngfw_as/ -j8 a-dnv-ipv6 ISBUILDRELEASE=1 NOTBUILDUNIWEBUI=1 HS_SHELL_PASSWORD=0 HS_BUILD_COVERAGE=0 HS_BUILD_COVERITY=0 OS_IMA
GE_FTP_IP=10.200.6.10 IMG_NAME=SG6000-RELR11-ADNV-V6-r0215-bzhao >build.log 2>&1"
================================================================================================================================================================================
Run make command under the project root, i.e. "/home/bzhao/repos/REL_R11"
```

2. `rua mkinfo -w A1000`：生成 A1000 平台的构建指令，启用 WebUI 支持

```
bzhao in build18-bj in ~/repos/REL_R11
> rua mkinfo -w A1000
1 matched info:
================================================================================================================================================================================
Product   : SG-6000-A1000
Model     : HS_TYPE_SG6000_A1000
Family    : HS_PRODUCT_FAMILY_FIREWALL
Platform  : HS_A1000
Target    : a-dnv
Directory : products/ngfw_as/
Command   : hsdocker7 "make -C products/ngfw_as/ -j8 a-dnv ISBUILDRELEASE=1 NOTBUILDUNIWEBUI=0 HS_SHELL_PASSWORD=0 HS_BUILD_COVERAGE=0 HS_BUILD_COVERITY=0 OS_IMAGE_FT
P_IP=10.200.6.10 IMG_NAME=SG6000-RELR11-ADNV-r0215-bzhao >build.log 2>&1"
================================================================================================================================================================================
Run make command under the project root, i.e. "/home/bzhao/repos/REL_R11"
```

3. `rua mkinfo -6w A1000`：生成 A1000 平台的构建指令，启用 IPv6 支持以及 WebUI 支持

```
bzhao in build18-bj in ~/repos/REL_R11
> rua mkinfo -6w A1000
1 matched info:
================================================================================================================================================================================
Product   : SG-6000-A1000
Model     : HS_TYPE_SG6000_A1000
Family    : HS_PRODUCT_FAMILY_FIREWALL
Platform  : HS_A1000
Target    : a-dnv-ipv6
Directory : products/ngfw_as/
Command   : hsdocker7 "make -C products/ngfw_as/ -j8 a-dnv-ipv6 ISBUILDRELEASE=1 NOTBUILDUNIWEBUI=0 HS_SHELL_PASSWORD=0 HS_BUILD_COVERAGE=0 HS_BUILD_COVERITY=0 OS_IMA
GE_FTP_IP=10.200.6.10 IMG_NAME=SG6000-RELR11-ADNV-V6-r0215-bzhao >build.log 2>&1"
================================================================================================================================================================================
Run make command under the project root, i.e. "/home/bzhao/repos/REL_R11"
```

4. `rua mkinfo -s s A1000`：生成 A1000 平台的构建指令，上传到 10.200.6.10 服务器，即苏州服务器

```
bzhao in build18-bj in ~/repos/REL_R11
> rua mkinfo -s s A1000
1 matched info:
================================================================================================================================================================================
Product   : SG-6000-A1000
Model     : HS_TYPE_SG6000_A1000
Family    : HS_PRODUCT_FAMILY_FIREWALL
Platform  : HS_A1000
Target    : a-dnv
Directory : products/ngfw_as/
Command   : hsdocker7 "make -C products/ngfw_as/ -j8 a-dnv ISBUILDRELEASE=1 NOTBUILDUNIWEBUI=1 HS_SHELL_PASSWORD=0 HS_BUILD_COVERAGE=0 HS_BUILD_COVERITY=0 OS_IMAGE_FT
P_IP=10.200.6.10 IMG_NAME=SG6000-RELR11-ADNV-r0215-bzhao >build.log 2>&1"
================================================================================================================================================================================
Run make command under the project root, i.e. "/home/bzhao/repos/REL_R11"
```

5. `rua mkinfo --format json A1000`：生成 A1000 平台的构建指令，输出格式指定为 JSON 格式，适合脚本使用

```
bzhao in build18-bj in ~/repos/REL_R11
> rua mkinfo --format json A1000
[
  {
    "Command": "hsdocker7 \"make -C products/ngfw_as/ -j8 a-dnv ISBUILDRELEASE=1 NOTBUILDUNIWEBUI=1 HS_SHELL_PASSWORD=0 HS_BUILD_COVERAGE=0 HS_BUILD_COVERITY=0 OS_IMA
GE_FTP_IP=10.200.6.10 IMG_NAME=SG6000-RELR11-ADNV-r0215-bzhao >build.log 2>&1\"",
    "Directory": "products/ngfw_as/",
    "Family": "HS_PRODUCT_FAMILY_FIREWALL",
    "Model": "SG-6000-A1000",
    "Platform": "HS_A1000",
    "Product": "SG-6000-A1000",
    "Target": "a-dnv"
  }
]
```

2 compdb

`rua compdb` 包含了众多子命令，分别用于生成、删除和管理编译数据库。编译数据库的存在是为了给 C/C++语言服务器（Language Server，LS），如 clangd，提供编译指示，从而使其能够在代码库中正确地跳转。

编译数据库包含了代码库中各个源文件的编译指令，有了该指令后，LS 就知道了该翻译单元的头文件查找路径和各种宏定义。通常而言，编译数据库是分构建目标的，如 `a-dnv-ipv6` 对应有一个编译数据库，`a-dnv` 对应有另一个编译数据库。

2.1 用法

用户可通过 `rua compdb -h` 查看帮助信息：

```
• [podman] ❱ ./target/x86_64-unknown-linux-musl/release/rua compdb -h
Manipulate compilation database

Usage: rua compdb <COMMAND>

Commands:
  gen     Generate a JSON compilation database (JCDB) for the given target
  add     Add the currently used compilation database into store as a new generation
  del     Delete compilation database generation(s) from store [aliases: rm]
  ls      List all compilation database generations in store
  use     Select a compilation database generation from store to use
  name    Name a compilation database generation
  remark  Remark a compilation database generation
  help    Print this message or the help of the given subcommand(s)

Options:
  -h, --help  Print help
```

compdb 包含七个子命令，分别是 `gen`，`add`，`del`，`ls`，`use`，`name`，`remark`。下面将分别介绍这些子命令的用法和示例。

2.1.1 生成编译数据库

`rua compdb gen <构建路径> <构建目标>`

- 构建路径：`products/ngfw_as` or `products/ngfw_ak` or …
- 构建目标：`a-dnv` or `hygon` or …

```
• [podman] ❱ ./target/x86_64-unknown-linux-musl/release/rua compdb gen -h
Generate a JSON compilation database (JCDB) for the given target

Usage: rua compdb gen [OPTIONS] <PATH> <TARGET>

Arguments:
  <PATH>    Path for the target where platform-specific makefiles reside, such as 'products/vfw'
  <TARGET>  Target to build, such as 'a-dnv'

Options:
  -D, --define <KEY=VAL>
          Define a variable which will be passed to the underlying make command
  -e, --engine <ENGINE>
          Engine for generating compilation database (defaults to built-in) [possible values: built-in, intercept-build, bear]
  -b, --bear-path <BEAR>
          Path to the bear binary (defaults to /devel/sw/bear/bin/bear)
  -i, --intercept-build-path <INTERCEPT-BUILD>
          Path to the intercept-build binary (defaults to /devel/sw/llvm/bin/intercept-build)
  -h, --help
          Print help (see more with '--help')

Examples:
  rua compdb gen products/ngfw_as a-dnv                  # For A1000/A2000...
  rua compdb gen products/ngfw_as a-dnv-ipv6             # For A1000/A2000... with IPv6 support
  rua compdb gen -e intercept-build products/ngfw_as a-dnv # For A1000/A2000... using intercept-build
  rua compdb gen . a-dnv                                 # For A1000/A2000... under submod dir
  rua compdb gen -e bear . a-dnv                         # For A1000/A2000... under submod dir using bear
  run compdb gen -e intercept-build . a-dnv              # For A1000/A2000... under submod dir using intercept-build

Caution:
  Some files are modified while running in built-in mode which is the default
  and faster:
  1. When running under project root dir:
     - scripts/last-rules.mk
     - scripts/rules.mk
     - Makefile
  2. When running under submod dir:
     - scripts/last-rules.mk
     - scripts/rules.mk
  These files may be left dirty if compdb process aborted unexpectedly. You
  could manually restore them by execute:
  svn revert Makefile scripts/last-rules.mk scripts/rules.mk
```

2.1.2 添加编译数据库

`rua compdb add <构建目标>`

- 构建目标：`a-dnv` or `hygon`…

```
• [podman] ❱ ./target/x86_64-unknown-linux-musl/release/rua compdb add -h
Add the currently used compilation database into store as a new generation

Usage: rua compdb add [OPTIONS] <TARGET>

Arguments:
  <TARGET>  Target for the compilation database

Options:
  -r, --revision <REVISION>
          Revision for compilation database (defaults to current repo revision)
  -f, --compilation-database <COMPILATION-DATABASE>
          Use this compilation database other than the default (compile_commands.json)
  -h, --help
          Print help
```

2.1.3 删除编译数据库

`rua compdb del <GENERATION-ID>`

```
• [podman] ❱ ./target/x86_64-unknown-linux-musl/release/rua compdb del -h
Delete compilation database generation(s) from store

Usage: rua compdb del [OPTIONS] [GENERATION-ID]

Arguments:
  [GENERATION-ID]  Generation to delete

Options:
  -a, --all      Remove all generations
  -n, --new <N>  Remove N newest generations
  -o, --old <N>  Remove N oldest generations
  -h, --help     Print help
```

2.1.4 列出编译数据库

`rua compdb ls`

```
• [podman] ❱ ./target/x86_64-unknown-linux-musl/release/rua compdb ls -h
List all compilation database generations in store

Usage: rua compdb ls

Options:
  -h, --help  Print help
```

2.1.5 选择编译数据库

`rua compdb use <GENERATION-ID>`

```
• [podman] ❱ ./target/x86_64-unknown-linux-musl/release/rua compdb use -h
Select a compilation database generation from store to use

Usage: rua compdb use <GENERATION>

Arguments:
  <GENERATION>  Compilation database generation id

Options:
  -h, --help  Print help
```

2.1.6 命名编译数据库

`rua compdb name <GENERATION-ID> <名字>`

```
• [podman] ❱ ./target/x86_64-unknown-linux-musl/release/rua compdb name -h
Name a compilation database generation

Usage: rua compdb name <GENERATION> <NAME>

Arguments:
  <GENERATION>  The compilation database generation
  <NAME>        Name for the compilation database

Options:
  -h, --help  Print help
```

2.1.7 备注编译数据库

`rua compdb remark <GENERATION-ID> <备注>`

```
• [podman] ❱ ./target/x86_64-unknown-linux-musl/release/rua compdb remark -h
Remark a compilation database generation

Usage: rua compdb remark <GENERATION> <REMARK>

Arguments:
  <GENERATION>  The compilation database generation
  <REMARK>      Remark for the compilation database generation

Options:
  -h, --help  Print help
```

2.2 示例

3 showcc

3.1 用法

3.2 示例

4 review

4.1 用法

4.2 示例

5 perfan

5.1 用法

5.2 示例

6 init

6.1 用法

6.2 示例