

Makefile **Sat Jan 13 01:11:58 2024** **1**

```
# Makefile front-end for rust dev works.
VERSION      = 1.0
RELEASE      = 2
DATE         = $(shell date)
NEWRELEASE   = $(shell echo $$(( $(RELEASE) + 1 )) )
PROJECT_NAME = rexpact-examples
TOPDIR       = $(shell pwd)
MANPAGES     =
A2PS2S1C     = /usr/bin/enscript -H1 --highlight-bar-gray=08 -fCourier8 -Ebash
#A2PS2S1C     = /usr/bin/enscript -H1 --highlight-bar-gray=08 -fconsolars7 -Ebash
A2PSTMP      = ./tmp
DOCS         = ./docs

SHELL := /bin/bash

.PHONY: all commit tests examples pdf doc docs sqld cmkrest cmd-demo cmk-agent-ctl cmkc

all: help
#https://stackoverflow.com/questions/6273608/how-to-pass-argument-to-makefile-from-command-line
args = `arg="$(filter-out $@,$(MAKECMDGOALS))" && echo ${arg:-${1}}`
%:
    @:

versionfile:
    echo "version:" $(VERSION) > etc/version
    echo "release:" $(RELEASE) >> etc/version
    echo "source build date:" $(DATE) >> etc/version

manpage:
    for manpage in $(MANPAGES); do (pod2man --center=$$manpage --release="" ./docs/$$manpage.pod > ./docs/$$manpage.1); done

clean: cleantmp
    -rm -rf *~
    -rm -rf docs/*.1
    -find . -type f -name *~ -exec rm -f {} \;
    -find . -type d -name target -exec rm -rf {} \;

clean_hard:
    -rm -rf $(shell $(PYTHON) -c "from distutils.sysconfig import get_python_lib; print get_python_lib()"
) /adagios

clean_hardest: clean_rpms

#Ref: https://stackoverflow.com/questions/1490949/how-to-write-loop-in-a-makefile
# MANIFEST
SRC1= Makefile Cargo.toml
SRC2= src/app.rs          src/bin/main.rs  src/controllers/guide.rs  src/controllers/home2.rs  src/controllers/home.rs  src/controllers/mod.rs  src/lib.rs  src/views/home.rs  src/views/mod.rs
SRC3= tests/requests/guide.rs  tests/requests/home2.rs  tests/requests/home.rs  tests/requests/mod.rs  tests/requests/snapshots/can_get_home2@home2_request.snap  tests/requests/snapshots/home2_request.snap  tests/requests/snapshots/can_get_home@home_request.snap
#SRC2= manage.py profiles_projects-dir-layout.txt

tmpdir:
    mkdir -p ${A2PSTMP}/src/bin ${A2PSTMP}/src/controllers ${A2PSTMP}/src/views ${A2PSTMP}/tests/requests/snapshots
cleantmp:
    rm -f ${A2PSTMP}/*.ps ${A2PSTMP}/*.pdf
.ps: cleantmp
    $(foreach var, $(SRC1), ${A2PS2S1C} --output=${A2PSTMP}/${(var)}.ps ${(var)} ;)
    $(foreach var, $(SRC2), ${A2PS2S1C} --output=${A2PSTMP}/${(var)}.ps ${(var)} ;)
    $(foreach var, $(SRC3), ${A2PS2S1C} --output=${A2PSTMP}/${(var)}.ps ${(var)} ;)
    touch .ps

allpdf: .pdf
touch .pdf
cp docs/src/*.pdf /mnt/hgfs/vmware

pdf: .pdf
touch .pdf
.pdf: .ps
    $(foreach var, $(SRC1), (cd ${A2PSTMP};ps2pdf ${(var)}.ps ${(var)}.pdf);)
    $(foreach var, $(SRC2), (cd ${A2PSTMP};ps2pdf ${(var)}.ps ${(var)}.pdf);)
    $(foreach var, $(SRC3), (cd ${A2PSTMP};ps2pdf ${(var)}.ps ${(var)}.pdf);)
```

```
rm -f ${A2PSTMP}/*.ps
cp ${A2PSTMP}/*.pdf ${DOCS}/src
rsync -azpv ${A2PSTMP}/src/* ${DOCS}/src
rsync -azpv ${A2PSTMP}/tests/* ${DOCS}/tests
find ${DOCS}/src/ -type f -name "*.ps" -exec rm -f {} \;
find ${DOCS}/src/ -type f
find ${DOCS}/tests/ -type f -name "*.ps" -exec rm -f {} \;
find ${DOCS}/tests/ -type f
touch .pdf
tree: clean
    tree -L 4 > ${PROJECT_NAME}-dir-layout.txt

.PHONY: tests test2
tests:
    (python -m unittest tests/test_cmdb.py)

# enable makefile to accept argument after command
#https://stackoverflow.com/questions/6273608/how-to-pass-argument-to-makefile-from-command-line

args = `arg="${(filter-out $@,$(MAKECMDGOALS))}" && echo ${arg:-${1}}`
%:
    @:

status:
    git status
commit:
    git commit -am "$(call args, Automated commit message without details, Please read the git diff)" &&
    git push
pull:
    git pull
install:
    cargo install --force --path .
#    cd lenovoad && cargo install --force --path .
#    cd 15_lettre && cargo install --force --path .
#    cd lettre && cargo install --force --path .
#    cd hello && cargo install --force --path .
#    cd wcr && cargo install --force --path .
#    cd headr && cargo install --force --path .
    ls -lrt ${HOME}/.cargo/bin

cmk-agent-ctl:
    (cd cmk-agent-ctl && cargo install --force --path .)
cmd-demo:
    (cd cmd-demo && ./cmd-install.sh)

cmkc:
    (cd cmkc && cargo install --force --path .)

cmkrest:
    (cd cmkrest && cargo install --force --path .)

install2:
#    cd sqld/sqlc && cargo install --force --path .
#    cd sqld/sqlc && cargo install --force --path .
    ls -lrt ${HOME}/.cargo/bin

installcmk:
    cd cmk && cargo install --force --path .
    ls -lrt ${HOME}/.cargo/bin | tail -10

installsqlc:
    cd sqld && make install
    ls -lrt ${HOME}/.cargo/bin | tail -10

clip:
    cargo clippy
build:
    cargo build && find target/debug -maxdepth 1 -type f -perm /755 | egrep -v "\.d"
sqlc:
    cd sqld && cargo build && find target/debug -maxdepth 1 -type f -perm /755 | egrep -v "\.d"
    cd sqld && cp -p target/debug/sqlc target/debug/bottomless-cli ${HOME}/.cargo/bin
    ls -lrt ${HOME}/.cargo/bin
buildad:
    cargo build -p lenovoad && find target/debug -maxdepth 1 -type f -perm /755 | egrep -v "\.d"
format:
    cargo fmt -v
```

examples:

```
cargo build --examples && ls -lrt target/debug/examples/ |egrep -v "\.d"
```

doc:

```
rustdoc README.md --crate-name docs
```

```
cargo doc
```

```
ls target/doc doc
```

```
# rustdoc src/bin/cmdmbc.rs --crate-name docs
```

test:

```
cargo test --bin cmdmbc -- --test-threads 1 --nocapture + 01 02 03 04 05 06 07 08 09 10 11 12 13 14
```

test2: build

```
@echo "..... cmdbclient help ....."
```

```
target/debug/cmdmbc --help
```

```
@echo "..... Run cmdmbc -s -h va32rocky8t03 ....."
```

```
target/debug/cmdmbc -s -h va32rocky8t03
```

help:

```
@echo "Usage: make <target> <argument>"
```

```
@echo
```

```
@echo "Available targets are:"
```

```
@echo " pdf Generate selected files in pdf and copy into ./docs"
```

```
@echo " allpdf Generate selected files in pdf and copy into vmware hgfs"
```

```
@echo " test build and test run"
```

```
@echo " format run cargo fmt to format the rust code"
```

```
@echo " build call up cargo build"
```

```
@echo " examples build all test programs in examples dir"
```

```
@echo " install install the binary in --path ."
```

```
@echo " help Showing this help "
```

```
@echo " clean clean all artifact files"
```

```
@echo " commit {"my message"} ie, git commit and push with defalut message"
```

```
@echo " status ie, git status"
```

```
@echo " pull ie, git pull"
```

```
@echo ""
```