QMake fundamentals

Useful links

Reference

Variables

Test functions

Replace functions

Language specification

QMake variables

Variable manipulation - general

Variable expansion

Simple access: variable types

```
# Simple variables - double-dollars
MY_VARIABLE = value1
message("My variable is: " $$MY_VARIABLE)

# Environment variables - soft brackets
message("Processor count: " $$(NUMBER_OF_PROCESSORS))

# QMake built-in properties - hard brackets
message($$[QT_INSTALL_PLUGINS])
```

Simple access: adding/removing values

```
MY_VARIABLE = VALUE1 # Sets a variable/value
!build_pass:message("MY_VARIABLE is: " $$MY_VARIABLE)

MY_VARIABLE -= VALUE1 # Removes a value
MY_VARIABLE += VALUE2 # Adds a value
MY_VARIABLE *= VALUE2 # Adds a value if it is not already set
MY_VARIABLE *= VALUE3

!build_pass:message("MY_VARIABLE has changed to: " $$MY_VARIABLE)
!build_pass:message($$first(MY_VARIABLE))
!build_pass:message($$last(MY_VARIABLE))

defined(MY_VARIABLE,var) # Returns true - second arg can be test (test
functions), replace (replace functions) or var. default is all functions.
unset(MY_VARIABLE) # Remove from current context
defined(MY_VARIABLE,var) # Returns false
```

Display

Prefix with !build_pass: to ensure a message appears only once (by default qmake is run per makefile created)

```
!build_pass:message("Simple message") # Prints to General Messages
!build_pass:warning("Warning message") # Prints to Issues
!build_pass:error("Error message") # Prints to Issues and exits qmake
!build_pass:log("Terminal log message") # Prints to Terminal
```

Variable manipulation - some specific use cases

Current QMake environment settings

```
!build_pass:message("Project working directory: " $$PWD)
!build_pass:message("Current project file: " $$_PRO_FILE_)
!build_pass:message("Plugin install diretory: " $$[QT_INSTALL_PLUGINS])
!build_pass:message("Variables: " $$enumerate_vars()) # All currently available
variables (can be used with contains)
```

Determining current build type

```
# Canonical example:
CONFIG(debug,debug|release){
    BUILD_TYPE = Debug
}
CONFIG(release,debug|release){
    BUILD_TYPE = Release
}

# Alternative
isActiveConfig(debug,debug|release){
    BUILD_TYPE = Debug
}else{
    BUILD_TYPE = Release
}
```

Naming

```
isActiveConfig(debug,debug|release)){
    message("Debug build")
    TARGET = wizardlibraryd
}else{
    message("Release build")
    TARGET = wizardlibrary
}
```

Variable inspection and scoping

```
contains(QT, sql){
    message("SQL module included")
    LIBS += -lodbc32
}
```

OpenMP

```
win32{
    QMAKE_CXXFLAGS += -openmp
    LIBS += -lvcomp
    message("OpenMP for win32")
}
else{
    QMAKE_CXXFLAGS += -fopenmp
    QMAKE_LFLAGS += -fopenmp
    LIBS += -lopenmp
    message("OpenMP for unix")
}
```

Build related stuff

```
message("Compiler: "$$QMAKE_CXX)
message("Architecture: "$$QMAKE_HOST.arch)
message("Building on: " $$QMAKE_HOST.name)
```

Environment variables (alternative method)

```
message("OS from environment: "$$getenv(OS))
```

Custom variables

```
MY_STATUS = newby
MY_STATUS += green
contains(MY_STATUS, newby){
    message("You're a newby")
    contains(MY_STATUS, green){
        message ("And you're so green you need mowing")
    }
}
isEmpty(MY_STATUS){
    message("MY_STATUS is not set")
}
```

Filesystem

Test for the existence of a file

```
exists("Copy-Binaries.ps1"){
    message("Copy-Binaries.ps1 exists")
}else{
    warning("Copy-Binaries.ps1 does not exist")
}
```

Create a directory

```
mkpath("new_dir")
exists("new_dir"){
    message("./new_dir exists")
}else{
    warning("./new_dir does not exist")
}
```

Cat a file

```
message($$cat("cat_test.txt",lines))
message($$cat("cat_test.txt",blob))
```

File names

```
FILE_NAME = "C:/temp/cat_test.txt"
message("Directory;" $$dirname(FILE_NAME))
message("Basename;" $$basename(FILE_NAME))
```

Find files

```
HPP_LIST = $$files("*.hpp", true)
for(a,HPP_LIST): message("Found a header: " $$a)
```

Miscellaneous

```
ANSWER = $$prompt("Give me something to work with:")
message("Response:" $$ANSWER)
```

A possible summary output

```
isActiveConfig(debug,debug|release){
    BUILD_TYPE = Debug
}else{
    BUILD_TYPE = Release
}

!build_pass:message($$BUILD_TYPE " - Project working directory: " $$PWD)
```

```
!build_pass:message($$BUILD_TYPE " - Directory contining current project file: "
$$_PRO_FILE_PWD_)
!build_pass:message($$BUILD_TYPE " - Current project file: " $$_PRO_FILE_)
!build_pass:message($$BUILD_TYPE " - Plugin installation directory: "
$$[QT_INSTALL_PLUGINS])
!build_pass:message($$BUILD_TYPE " - Compiler: " $$QMAKE_CXX)
!build_pass:message($$BUILD_TYPE " - Architecture: " $$QMAKE_HOST.arch)
!build_pass:message($$BUILD_TYPE " - Building on: " $$QMAKE_HOST.name)
!build_pass:message($$BUILD_TYPE " - Processor count: " $$(NUMBER_OF_PROCESSORS))

# Check for OpenMP and odbc32 linking
contains(QT,sql){
    LIBS += -lodbc32
}

OPENMP = False
contains(LIBS,-lopenmp){
    OPENMP = MinGW
}
contains(LIBS,-lvcomp){
    OPENMP = MSVC
}
!build_pass:message($$BUILD_TYPE " - Using OpenMP: " $$OPENMP)

ODBC32 = False
contains(LIBS,-lodbc32){
    ODBC32 = True
}
!build_pass:message($$BUILD_TYPE " - Linking to odbc32: " $$ODBC32)
```