Bitbake Cheat Sheet

Command Line options

-c <task >	execute <task >for the image or recipe being built. ex: bitbake -c fetch busybox. Some of the possible tasks are: fetch, configure, compile, package, clean
-f	force execution of the operation, even if not required
-v	show verbose output
-DDD	show lots of debug information
-s	show recipe version information
–help	get usage help
-c listtasks <image-or-recipe-name >	show the tasks associated with an image or individual recipe
-g <recipe >	output dependency tree in graphviz format
-e	Show the global or per-package environment

Useful commands

bitbake foo	this builds package foo
bitbake -c clean foo	This command will clean up your tmp dir for the given package. It is very useful if you work on a new .bb recipe. Without it your changes to the recipe may not work.
bitbake -f -c compile foo	To recompile your source code if you change a line in it.
bitbake -s | grep foo	Check the version of the recipe
bitbake -c devshell <target >	opens the interactive devshell in the source directory of target, this is usefull for debugging build failures
bitbake virtual/kernel -c menuconfig	Interactive kernel configuration
bitbake <image >-c fetchall	Fetch all sources for that particular image. This should be execute before a long running build
bitbake -e <target >| grep SRC_REV	Check the content of any variable the recipe *target*

.bb file syntax

VAR = “foo”	simple assignment
VAR ?= “foo”	assign if no other value is already assigned (default assignment)
VAR ??=“foo”	weak default assignment, takes lower precedence than ?=
VAR = “stuff ${OTHER_VAR} more”	variable expansion, OTHER_VAR expanded at time of reference to VAR
VAR := “stuff ${OTHER_VAR} more”	immediate variable expansion, OTHER_VAR expanded at time of parsing this line
VAR += “foo”	append with space
VAR =+ “foo”	prepend with space
VAR .= “foo”	append without space
VAR =. “foo”	prepend without space
VAR_append = “foo”	append without space
VAR = “foo ${@<line-of-python-code >}”	python code expansion, ex: VAR = “the date is: ${@time.strftime(‘%Y%m%d’,time.gmtime())}”
include foo	include file, include file named “foo”, search BBPATH
require [<path>]foo	require file, include file named “foo”, failing if not found exactly where specified
inherit foo	inherit classes, include definitions from foo.bbclass
do_sometask() { <shell code>}	define a task using shell code
python do_sometask { <python code>}	define a task using python code
addtask sometask (before|after) other_task	add a task, adds a defined task to the list of tasks, with the ordering specified. Zero or more “before” or “after” clauses can be used.
VAR[some_flag]=“foo”	associate a subsidiary flag value to a variable, a few subsidiary flag value names are well-defined: “dirs”, “cleandirs”, “noexec”, “nostamp”, “fakeroot”, “umask”, “deptask”, “rdeptask”, “recdeptask”, “recrdeptask” Flag values appear to be used exclusively with task definitions (i.e. do_sometask)

Useful variables

INHERIT += rm_work	Removes source and temporary files after a successfull build this saves disk space.