```
###
# This makefile is used to generate the kernel documentation,
# primarily based on in-line comments in various source files.
# See Documentation/kernel-doc-nano-HOWTO.txt for instruction in how
# to document the SRC - and how to read it.
# To add a new book the only step required is to add the book to the
# list of DOCBOOKS.

DOCBOOKS := z8530book.xml mcabook.xml device-drivers.xml \
            kernel-hacking.xml kernel-locking.xml deviceiobook.xml \
            writing_usb_driver.xml networking.xml \
            kernel-api.xml filesystems.xml lsm.xml usb.xml kgdb.xml \
            gadget.xml libata.xml mtdnand.xml librs.xml rapidio.xml \
            genericirq.xml s390-drivers.xml uio-howto.xml scsi.xml \
            mac80211.xml debugobjects.xml sh.xml regulator.xml \
            alsa-driver-api.xml writing-an-alsa-driver.xml \
            tracepoint.xml media.xml drm.xml

###
# The build process is as follows (targets):
#               (xmldocs) [by docproc]
# file.tmpl --> file.xml +--> file.ps   (psdocs)   [by db2ps or xmlto]
#                        +--> file.pdf  (pdfdocs)  [by db2pdf or xmlto]
#                        +--> DIR=file  (htmldocs) [by xmlto]
#                        +--> man/      (mandocs)  [by xmlto]


# for PDF and PS output you can choose between xmlto and docbook-utils tools
PDF_METHOD      = $(prefer-db2x)
PS_METHOD       = $(prefer-db2x)


###
# The targets that may be used.
PHONY += xmldocs sgmldocs psdocs pdfdocs htmldocs mandocs installmandocs
cleandocs xmldoclinks

BOOKS := $(addprefix $(obj)/,$(DOCBOOKS))
xmldocs: $(BOOKS) xmldoclinks
sgmldocs: xmldocs

PS := $(patsubst %.xml, %.ps, $(BOOKS))
psdocs: $(PS)

PDF := $(patsubst %.xml, %.pdf, $(BOOKS))
pdfdocs: $(PDF)

HTML := $(sort $(patsubst %.xml, %.html, $(BOOKS)))
htmldocs: $(HTML)
        $(call build_main_index)
        $(call build_images)

MAN := $(patsubst %.xml, %.9, $(BOOKS))
mandocs: $(MAN)

build_images = mkdir -p $(objtree)/Documentation/DocBook/media/ && \
```

```
                cp $(srctree)/Documentation/DocBook/dvb/*.png
$(srctree)/Documentation/DocBook/v4l/*.gif
$(objtree)/Documentation/DocBook/media/

xmldoclinks:
ifneq ($(objtree),$(srctree))
        for dep in dvb media-entities.tmpl media-indices.tmpl v4l; do \
                rm -f $(objtree)/Documentation/DocBook/$$dep \
                && ln -s $(srctree)/Documentation/DocBook/$$dep
$(objtree)/Documentation/DocBook/ \
                || exit; \
        done
endif

installmandocs: mandocs
        mkdir -p /usr/local/man/man9/
        install Documentation/DocBook/man/*.9.gz /usr/local/man/man9/

###
#External programs used
KERNELDOC = $(srctree)/scripts/kernel-doc
DOCPROC   = $(objtree)/scripts/basic/docproc

XMLTOFLAGS = -m $(srctree)/Documentation/DocBook/stylesheet.xsl
XMLTOFLAGS += --skip-validation

###
# DOCPROC is used for two purposes:
# 1) To generate a dependency list for a .tmpl file
# 2) To preprocess a .tmpl file and call kernel-doc with
#     appropriate parameters.
# The following rules are used to generate the .xml documentation
# required to generate the final targets. (ps, pdf, html).
quiet_cmd_docproc = DOCPROC $@
      cmd_docproc = SRCTREE=$(srctree)/ $(DOCPROC) doc $< >$@
define rule_docproc
        set -e;                                                         \
        $(if $($(quiet)cmd_$(1)),echo '  $($(quiet)cmd_$(1))';)         \
        $(cmd_$(1));                                                    \
        (                                                               \
          echo 'cmd_$@ := $(cmd_$(1))';                                 \
          echo $@: `SRCTREE=$(srctree) $(DOCPROC) depend $<`;           \
        ) > $(dir $@).$(notdir $@).cmd
endef

%.xml: %.tmpl FORCE
        $(call if_changed_rule,docproc)

###
#Read in all saved dependency files
cmd_files := $(wildcard $(foreach f,$(BOOKS),$(dir $(f)).$(notdir $(f)).cmd))

ifneq ($(cmd_files),)
  include $(cmd_files)
endif
```

```
###
# Changes in kernel-doc force a rebuild of all documentation
$(BOOKS): $(KERNELDOC)

# Tell kbuild to always build the programs
always := $(hostprogs-y)

notfoundtemplate = echo "*** You have to install docbook-utils or xmlto ***"; \
                   exit 1
db2xtemplate = db2TYPE -o $(dir $@) $<
xmltotemplate = xmlto TYPE $(XMLTOFLAGS) -o $(dir $@) $<

# determine which methods are available
ifeq ($(shell which db2ps >/dev/null 2>&1 && echo found),found)
        use-db2x = db2x
        prefer-db2x = db2x
else
        use-db2x = notfound
        prefer-db2x = $(use-xmlto)
endif
ifeq ($(shell which xmlto >/dev/null 2>&1 && echo found),found)
        use-xmlto = xmlto
        prefer-xmlto = xmlto
else
        use-xmlto = notfound
        prefer-xmlto = $(use-db2x)
endif

# the commands, generated from the chosen template
quiet_cmd_db2ps = PS      $@
      cmd_db2ps = $(subst TYPE,ps, $($(PS_METHOD)template))
%.ps : %.xml
        $(call cmd,db2ps)

quiet_cmd_db2pdf = PDF     $@
      cmd_db2pdf = $(subst TYPE,pdf, $($(PDF_METHOD)template))
%.pdf : %.xml
        $(call cmd,db2pdf)


index = index.html
main_idx = Documentation/DocBook/$(index)
build_main_index = rm -rf $(main_idx) && \
                   echo '<h1>Linux Kernel HTML Documentation</h1>' >>
$(main_idx) && \
                   echo '<h2>Kernel Version: $(KERNELVERSION)</h2>' >>
$(main_idx) && \
                   cat $(HTML) >> $(main_idx)

quiet_cmd_db2html = HTML    $@
      cmd_db2html = xmlto xhtml $(XMLTOFLAGS) -o $(patsubst %.html,%,$@) $< && \
                echo '<a HREF="$(patsubst %.html,%,$(notdir $@))/index.html"> \
        $(patsubst %.html,%,$(notdir $@))</a><p>' > $@

%.html: %.xml
        @(which xmlto > /dev/null 2>&1) || \
```

```
         (echo "*** You need to install xmlto ***"; \
          exit 1)
        @rm -rf $@ $(patsubst %.html,%,$@)
        $(call cmd,db2html)
        @if [ ! -z "$(PNG-$(basename $(notdir $@)))" ]; then \
            cp $(PNG-$(basename $(notdir $@))) $(patsubst %.html,%,$@); fi

quiet_cmd_db2man = MAN     $@
      cmd_db2man = if grep -q refentry $<; then xmlto man $(XMLTOFLAGS) -o
$(obj)/man $< ; gzip -f $(obj)/man/*.9; fi
%.9 : %.xml
        @(which xmlto > /dev/null 2>&1) || \
         (echo "*** You need to install xmlto ***"; \
          exit 1)
        $(Q)mkdir -p $(obj)/man
        $(call cmd,db2man)
        @touch $@

###
# Rules to generate postscripts and PNG images from .fig format files
quiet_cmd_fig2eps = FIG2EPS $@
      cmd_fig2eps = fig2dev -Leps $< $@

%.eps: %.fig
        @(which fig2dev > /dev/null 2>&1) || \
         (echo "*** You need to install transfig ***"; \
          exit 1)
        $(call cmd,fig2eps)

quiet_cmd_fig2png = FIG2PNG $@
      cmd_fig2png = fig2dev -Lpng $< $@

%.png: %.fig
        @(which fig2dev > /dev/null 2>&1) || \
         (echo "*** You need to install transfig ***"; \
          exit 1)
        $(call cmd,fig2png)

###
# Rule to convert a .c file to inline XML documentation
       gen_xml = :
 quiet_gen_xml = echo '  GEN     $@'
silent_gen_xml = :
%.xml: %.c
        @$($(quiet)gen_xml)
        @(                              \
           echo "<programlisting>";     \
           expand --tabs=8 < $< |       \
           sed -e "s/&/\\&amp;/g"       \
               -e "s/</\\&lt;/g"        \
               -e "s/>/\\&gt;/g";       \
           echo "</programlisting>")  > $@

###
# Help targets as used by the top-level makefile
dochelp:
```

```
        @echo  ' Linux kernel internal documentation in different formats:'
        @echo  '  htmldocs        - HTML'
        @echo  '  pdfdocs         - PDF'
        @echo  '  psdocs          - Postscript'
        @echo  '  xmldocs         - XML DocBook'
        @echo  '  mandocs         - man pages'
        @echo  '  installmandocs  - install man pages generated by mandocs'
        @echo  '  cleandocs       - clean all generated DocBook files'

###
# Temporary files left by various tools
clean-files := $(DOCBOOKS) \
        $(patsubst %.xml, %.dvi,  $(DOCBOOKS)) \
        $(patsubst %.xml, %.aux,  $(DOCBOOKS)) \
        $(patsubst %.xml, %.tex,  $(DOCBOOKS)) \
        $(patsubst %.xml, %.log,  $(DOCBOOKS)) \
        $(patsubst %.xml, %.out,  $(DOCBOOKS)) \
        $(patsubst %.xml, %.ps,   $(DOCBOOKS)) \
        $(patsubst %.xml, %.pdf,  $(DOCBOOKS)) \
        $(patsubst %.xml, %.html, $(DOCBOOKS)) \
        $(patsubst %.xml, %.9,    $(DOCBOOKS)) \
        $(index)

clean-dirs := $(patsubst %.xml,%,$(DOCBOOKS)) man

cleandocs:
        $(Q)rm -f $(call objectify, $(clean-files))
        $(Q)rm -rf $(call objectify, $(clean-dirs))

# Declare the contents of the .PHONY variable as phony.  We keep that
# information in a variable se we can use it in if_changed and friends.

.PHONY: $(PHONY)
```