```
**********************************************************
    9563 Thu Sep 13 19:12:27 2018
new/usr/src/tools/scripts/Install.1onbld
9803 pbchk could use a -c option
9825 pbchk -b option should be -p
**********************************************************
.\"
.\" Copyright 2010 Sun Microsystems, Inc.  All rights reserved.
.\" Use is subject to license terms.
.\"
.\" CDDL HEADER START
.\"
.\" The contents of this file are subject to the terms of the
.\" Common Development and Distribution License (the "License").
.\" You may not use this file except in compliance with the License.
.\"
.\" You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
.\" or http://www.opensolaris.org/os/licensing.
.\" See the License for the specific language governing permissions
.\" and limitations under the License.
.\"
.\" When distributing Covered Code, include this CDDL HEADER in each
.\" file and include the License file at usr/src/OPENSOLARIS.LICENSE.
.\" If applicable, add the following below this CDDL HEADER, with the
.\" fields enclosed by brackets "[]" replaced with your own identifying
.\" information: Portions Copyright [yyyy] [name of copyright owner]
.\"
.\" CDDL HEADER END
.\"
.\" Copyright 2018 Joyent, Inc.
.\"
.TH INSTALL 1ONBLD "Jan 14, 2010"
.SH NAME
Install \- install a kernel from an ON workspace
.SH SYNOPSIS
.TP 8n
.B Install
.RB [ " \-w "
.IR workspace " ]"
.RB [ " \-s "
.IR "source dir" " ]"
.br
.RB [ " \-k  "
.IR "kernel arch" " ]"
.RB "[ " \-n " | " \-t|T
.IR target " ]"
.br
.RB [ " \-u|m|a " ]
.RB [ " \-v|V|q " ]
.RB [ " \-c|p " ]
.br
.RB [ " \-l "
.IR "library file" " ]"
.RB [ " \-L " ]
.RB [ " \-3 " ]
.RB [ " \-6 " ]
.RB [ " \-K " ]
.br
.RB [ " \-o "
{
.BR obj " | "
.B debug
}
]
.RB [ " \-d "
.IR "work dir" " ]"
```

```
.br
.RB [ " \-D "
.IR "library dir" " ]"
.RB [ " \-G "
.IB glomname " ]"
.RI [ " module ... " ]
.LP
or
.LP
.BR "Install \-R " "[ options ]"
.SH DESCRIPTION
.LP
.B Install
is a utility which simplifies the process of installing a 5.0 system.
.B Install
goes into a built ON workspace (or any kernel source tree),
looks at the Makefiles,
and figures out how to construct the /kernel and /usr/kernel directories.
It then creates a tarfile
.RB "(see " tar "(1))"
containing /kernel, /usr/kernel, and a few related /etc files.  If a
.I target ([user@]machine:/dir)
is specified, the tarfile is either copied to
.IR machine:/dir " (-T) or untarred on " "machine" " in " "/dir" " (-t),"
using the remote user id
.IR user ,
if specified.
With no options,
.B Install
creates a sun4c system from files in the current workspace (as indicated
by $SRC) and places the tarfile in /tmp/Install.username/Install.sun4c.tar.

.SH OPTIONS
.TP 20n
.BI "-w" " ws"
Install the system built in the ON workspace
.I ws.  ws
must be a built ON workspace \(em
.B Install
will not automatically invoke
.BR make (1) .
If \-w is not specified,
.B Install
uses the current
workspace (as indicated by $CODEMGR_WS).  If there is no current workspace,
.B Install
checks to see if you are in an appropriate source directory, e.g. uts/sun4c;
if so,
.B Install
takes files from there.  Otherwise,
.B Install
looks for files under $SRC/uts.
.TP
.BI "-s" " source directory"
where to look for files [default: $SRC/uts].
.TP
.BI "-k" " kernel arch"
the type of kernel to install.  The default is sun4c; however, if you invoke
.B Install
from $SRC/uts/sun4z,
.B Install
assumes you want a sun4z kernel.
.TP
.B "-n"
No target; just create the tarfile in
/tmp/Install.username/Install.sun4c.tar [default].
```

```
.BR "-n" " implies " "-p" .
.TP
.BI "-t" " target"
Install the system on
.I target ([user@]machine:/dir).
This means that kernel/unix is copied to
.I machine:/dir/kernel/unix,
etc.
.IR /dir " is typically either " / " or " /mnt.
.BR "-t" " implies " "-c" .
The default remote user id is the same as the local one ($LOGNAME).
.TP
.BI "-T" " target"
Copy the tarfile to
.I target ([user@]machine:/dir).
This creates the file
.I /dir/Install.tar
on
.I machine.
To finish the install, log on to
.I machine
as root, and type
.RB ‘‘ "cd /; tar xvf /dir/Install.tar" "’’."
.BR "-T" " implies " "-c" .
.TP
.B "-u"
Install unix only.
.TP
.B "-m"
Install modules only.
.TP
.B "-a"
Install unix and all modules [default].
.TP
.B "-v"
Verbose mode.
.TP
.B "-V"
REALLY verbose mode.  Useful mainly for debugging.
.TP
.B "-q"
Quiet mode [default].  Only fatal messages are printed.
.TP
.B "-c"
Clean up.  After a successful install, delete the files created in
/tmp/Install.username.  This is the default behavior if a
.I target
is specified with
.BR "-t" " or " "-T" .
.TP
.B "-p"
Preserve temp files.  This is the default behavior when no
.I target
is specified
.RB ( "-n" ).
.TP
.B "-R"
Recover from a failed
.BR Install .
This is not required, it’s just faster than restarting.
A typical scenario is for
.B Install
to run smoothly right up to the very end, but then die with
"Permission denied" when it tries to rsh/rcp to the target machine.
At this point, you log on to the target machine, diddle the permissions,
log off, and type
```

```
.RB ‘‘ "Install -R" "’’."
.B Install
will only have to retry the rsh/rcp,
rather than rebuild the tarfile from scratch.
.TP
.BI "-d" " temp directory"
specifies where
.B Install
should create its temp files [default: /tmp/Install.username].  This is
useful if you have limited space in /tmp (\fBInstall\fR can take as
much as 100MB).
The suffix "Install.username" is always appended.
.TP
.B "-L"
add a system to your library.  This allows you to build a personal
collection of installable systems from various environments and for
various architectures.  When you type
.RB ‘‘ "Install -w /ws/ws_name -k arch -L" "’’, " Install
creates a tarfile called
.I ws_name.arch.tar
in your library directory (~/LibInstall by default).
.BR "-L" " implies " "-c" .
.TP
.BI "-l" " library file"
Installs the system contained in
.I library file.
You may omit the ‘‘.tar’’ suffix.  For example,
.RB ‘‘ "Install -l my_ws.sun4c -t machine:/" ’’
installs a system you previously built with
.B "-L"
(from sun4c files in my_ws) on
.IR machine:/ .
This is equivalent to typing
.RB ‘‘ "rsh machine ’(cd /; tar xvf -)’ <~/LibInstall/my_ws.sun4c.tar" ’’,
but it’s easier to remember.
.TP
.BI "-D" " lib directory"
specifies the library directory [default: $HOME/LibInstall].
.TP
.BI "-G " glomname
gloms /kernel and /usr/kernel together into a single /kernel directory.
Useful for development work, e.g. use "Install -G good [...]" to create a
"/kernel.good".
.TP
.BR "-o " "{ \fBobj\fP | \fBdebug\fP }"
object directory. The default is "debug".
.TP
.B \-3
32-bit modules only
.TP
.B \-6
64-bit modules only
.TP
.B \-K
Do not include kmdb misc module or dmods
.TP
.B "-h"
Help.  Prints a brief summary of
.BR Install "’s"
options.
.LP
If you are in a directory like $SRC/uts/sun4z when you invoke
.BR Install ,
it will infer that you want to install a sun4z system
from the current workspace.
.LP
```

```
If you supply a list of modules, it overrides any of the
.B "-uma"
options.  You only need to specify the basename of the
module(s), e.g. ‘‘\fBInstall ufs nfs le\fR’’.
‘‘\fBInstall unix\fR’’ is equivalent to ‘‘\fBInstall -u\fR’’, and
‘‘\fBInstall modules\fR’’ is equivalent to ‘‘\fBInstall -m\fR’’.
.LP
You can customize
.B Install
by creating a .Installrc file in your home directory.  .Installrc
should consist of a list of command-line-style options, e.g:
.LP
.nf
.B
        -w /ws/foo
.fi
.br
.nf
.B
        -t labmachine:/mnt -pv
.fi
.LP
.B Install
processes default options first, then .Installrc
options, then command-line options.  In the case of
conflicting options (e.g. \fB-uma\fR), the last one wins.
.LP
In order to use the most convenient form of
.BR Install " (‘‘" "Install -t machine:/" "’’),"
you will need to do the following on the target machine:
.LP
.br
.nf
        (1) add your machine name to the /etc/hosts.equiv file
.fi
.br
.nf
        (2) add your username to the /etc/{passwd,shadow} files
.fi
.br
.nf
        (3) chown -R yourself /kernel /usr/kernel
.fi
.br
.nf
        (4) chmod -R u+w /kernel /usr/kernel
.fi
.SH "ENVIRONMENT"
.LP
You can set the following variables in your environment:
.LP
INSTALL_RC [default: $HOME/.Installrc]
.IP
file containing default options for \fBInstall\fR
.LP
INSTALL_STATE [default: $HOME/.Install.state]
.IP
where \fBInstall\fR keeps its state information
.LP
INSTALL_DIR [default: /tmp/Install.username]
.IP
where \fBInstall\fR does its work.  This can be overridden on
the command line with \fB\-d\fR.
.LP
INSTALL_LIB [default: $HOME/LibInstall]
.IP
```

```
where \fBInstall\fR gets/puts library files.  This can be overridden on
the command line with \fB\-D\fR.
.LP
INSTALL_CP [default: cp -p]
.IP
the command to copy files locally
.LP
INSTALL_RCP [default: rcp -p]
.IP
the command to copy files remotely
.SH "EXAMPLES"
.LP
.B
Install -w /ws/blort -t machine:/
.IP
.RI "installs the system built in workspace " /ws/blort " on " machine:/
.LP
.B
Install -w /ws/blort -T machine:/tmp
.br
.B
rsh machine -l root "cd /; tar xvf /tmp/Install.tar"
.IP
is an equivalent way to do the previous example
.LP
.B Install
.IP
makes a tarfile containing a sun4c kernel,
and places it in /tmp/Install.username/Install.sun4c.tar.  However, if you
are in one of the arch directories (e.g. $SRC/uts/sun4m) when you invoke
.BR Install ,
you will get a tarfile for that architecture instead.
.LP
.B
Install -k sun4m -w /ws/on493 -t mpbox:/ ufs
.IP
installs a new sun4m ufs module from workspace /ws/on493 on mpbox:/
.SH "FILES"
$HOME/.Installrc, $HOME/.Install.state
.SH "SEE ALSO"
.BR tar "(1), " rsh "(1), " rcp "(1)"
.SH "BUGS"
.BR tar "(1) and " rsh "(1)"
do not have particularly useful exit codes.  To compensate,
.B Install
feeds stderr through grep -v and throws away error messages which it
considers harmless.  If there’s anything left,
.B Install
assumes it is fatal.  It’s a hack, but it works.
```

```
**********************************************************
    4118 Thu Sep 13 19:12:28 2018
new/usr/src/tools/scripts/Makefile
9803 pbchk could use a -c option
9825 pbchk -b option should be -p
**********************************************************
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
#
# Copyright (c) 1999, 2010, Oracle and/or its affiliates. All rights reserved.
#
# Copyright 2010, Richard Lowe
#
# Copyright 2018 Joyent, Inc.

SHELL=/usr/bin/ksh93

SHFILES= \
        Install \
        bldenv \
        build_cscope \
        bringovercheck \
        checkpaths \
        cstyle \
        elfcmp \
        flg.flp \
        genoffsets \
        nightly \
        onu \
        protocmp.terse \
        sccscheck \
        webrev \
        which_scm \
        ws \
        xref

PERLFILES= \
        check_rtime \
        find_elf \
        interface_check \
        interface_cmp \
        jstyle \
        validate_flg \
        validate_paths \
        wdiff

PERLMODULES= \
        onbld_elfmod.pm \
```

```
        onbld_elfmod_vertype.pm


PYFILES= \
        cddlchk \
        copyrightchk \
        git-pbchk \
        hdrchk \
        mapfilechk \
        validate_pkg \
        wscheck \
        wsdiff

SCRIPTLINKS= \
        git-nits

MAN1ONBLDFILES= \
        Install.1onbld \
        bldenv.1onbld \
        bringovercheck.1onbld \
        cddlchk.1onbld \
        checkpaths.1onbld \
        check_rtime.1onbld \
        cstyle.1onbld \
        find_elf.1onbld \
        flg.flp.1onbld \
        git-pbchk.1onbld \
        hdrchk.1onbld \
        interface_check.1onbld \
        interface_cmp.1onbld \
        jstyle.1onbld \
        mapfilechk.1onbld \
        nightly.1onbld \
        onu.1onbld \
        sccscheck.1onbld \
        webrev.1onbld \
        which_scm.1onbld \
        ws.1onbld \
        wsdiff.1onbld \
        xref.1onbld

MAN1ONBLDLINKS= \
        git-nits.1onbld

MAKEFILES= \
        xref.mk

ETCFILES= \
        its.conf \
        its.reg

EXCEPTFILES= \
        check_rtime \
        interface_check \
        interface_cmp

CLEANFILES = $(SHFILES) $(PERLFILES) $(PYFILES) bldenv.1onbld onu.sh

onu.sh: onu.sh.in
        $(SED) -e "s:@PYTHON_VERSION@:$(PYTHON_VERSION):g" < onu.sh.in > $@

include ../Makefile.tools

ROOTONBLDSCRIPTLINKS = $(SCRIPTLINKS:%=$(ROOTONBLDBIN)/%)
ROOTONBLDMAN1ONBLDLINKS = $(MAN1ONBLDLINKS:%=$(ROOTONBLDMAN1ONBLD)/%)
```

```
$(ROOTONBLDETCFILES)            := FILEMODE=    644
$(ROOTONBLDEXCEPTFILES)         := FILEMODE=    644
$(ROOTONBLDPERLMODULES)         := FILEMODE=    644
$(ROOTONBLDMAKEFILES)           := FILEMODE=    644
$(ROOTONBLDMAN1ONBLDFILES)      := FILEMODE=    644

.KEEP_STATE:

all:    $(SHFILES) $(PERLFILES) $(PERLMODULES) $(PYFILES) \
        $(MAN1ONBLDFILES) $(MAKEFILES)

onu.sh: onu.sh.in
        $(SED) -e "s:@PYTHON_VERSION@:$(PYTHON_VERSION):g" < onu.sh.in > $@

$(ROOTONBLDBIN)/git-nits:
        $(RM) $(ROOTONBLDBIN)/git-nits
        $(SYMLINK) git-pbchk $(ROOTONBLDBIN)/git-nits

$(ROOTONBLDMAN1ONBLD)/git-nits.1onbld:
        $(RM) $(ROOTONBLDMAN1ONBLD)/git-nits.1onbld
        $(SYMLINK) git-pbchk.1onbld $(ROOTONBLDMAN1ONBLD)/git-nits.1onbld

install: all .WAIT $(ROOTONBLDSHFILES) $(ROOTONBLDPERLFILES)            \
                $(ROOTONBLDPERLMODULES) $(ROOTONBLDPYFILES)             \
                $(ROOTONBLDSCRIPTLINKS) $(ROOTONBLDMAN1ONBLDFILES)      \
                $(ROOTONBLDMAKEFILES) $(ROOTONBLDETCFILES)              \
                $(ROOTONBLDEXCEPTFILES) $(ROOTONBLDMAN1ONBLDLINKS)

clean:
        $(RM) $(CLEANFILES)

bldenv: bldenv.sh stdenv.sh
        $(RM) "$@"
        sed -e '/# STDENV_START/ r stdenv.sh' bldenv.sh > "$@"
        # Check for shell lint and fail if we hit warnings
        shlintout="$$( /usr/bin/ksh93 -n "$@" 2>&1 )" ; \
                [[ "$${shlintout}" != "" ]] && \
                { print -r -- "$${shlintout}" ; false ; } || true
        $(CHMOD) +x "$@"

bldenv.1onbld: bldenv
        $(RM) "$@"
        (set +o errexit ; ksh93 $? --nroff ; true) 2>&1 | \
        sed -e 's/\.DS/.nf/g;s/\.DE/.fi/' \
        -e 's/\.TH BLDENV 1/.TH BLDENV 1ONBLD "September 4, 2018"/' \
        -e 's/.OP \([a-z]\) - flag -/.OP \\-\1/g' \
        -e 's/\.TH BLDENV 1/.TH BLDENV 1ONBLD/' \
        -e 's/(1)/(1ONBLD)/' > "$@"

nightly: nightly.sh stdenv.sh
        $(RM) "$@"
        sed -e '/# STDENV_START/ r stdenv.sh' nightly.sh > nightly
        $(CHMOD) +x "$@"

#
# Not run by default: bootstrap...
check:
        $(ROOTONBLDBINMACH)/mandoc -Tlint -Wwarning $(MAN1ONBLDFILES)

include ../Makefile.targ
```

```
**********************************************************
    3193 Thu Sep 13 19:12:28 2018
new/usr/src/tools/scripts/git-pbchk.1onbld
9803 pbchk could use a -c option
9825 pbchk -b option should be -p
**********************************************************
   1 '\" t
   2 .\"
   3 .\" This file and its contents are supplied under the terms of the
   4 .\" Common Development and Distribution License ("CDDL"), version 1.0.
   5 .\" You may only use this file in accordance with the terms of version
   6 .\" 1.0 of the CDDL.
   7 .\"
   8 .\" A full copy of the text of the CDDL should have accompanied this
   9 .\" source.  A copy of the CDDL is also available via the Internet at
  10 .\" http://www.illumos.org/license/CDDL.
  11 .\"
  12 .\"
  13 .\" Copyright 2011 Richard Lowe.
  14 .\" Copyright 2015 Elysium Digital, L.L.C.
  15 .\" Copyright 2018 Joyent, Inc.
  16 .\"

  18 .TH "GIT\-PBCHK" "1ONBLD" "September 4, 2018" "" ""
  17 .TH "GIT\-PBCHK" "1ONBLD" "April 23, 2015" "" ""

  20 .SH "NAME"
  21 \fBgit\-pbchk\fR \- nits and pre\-putback checks for git

  23 .SH "SYNOPSIS"
  24 git\-pbchk [\-c \fIcheck\fR] [\-p \fIbranch\fR] [file...]
  23 git\-pbchk [\-b \fIbranch\fR]

  26 .P
  27 git\-nits [\-c \fIcheck\fR] [\-p \fIbranch\fR] [file...]
  26 git\-nits [\-b \fIbranch\fR]

  29 .SH "OPTIONS"

  31 .TP
  32 \fB\-c check\fR:
  33 .IP
  34 Run the specific \fIcheck\fR, as named below.
  35 In this mode, individual files can be provided to check.
  36 .TP
  37 \fB\-p branch\fR:
  38 .IP
  39 Compare the current workspace to the parent \fIbranch\fR for the purposes of gen
  40 .IP
  41 If this option is not specified an attempt is made to determine this automatical
  42 .IP
  43 If no branch is specified and none can be determined automatically \fBorigin/mas
  44 .SH "DESCRIPTION"
  45 Check your workspace for common nits and putback\-ending mistakes, a simple set

  46 .TP
  47 Comment format [comchk]
  32 Comment format
  48 Check that putback comments follow the prescribed format (only run for pbchk)

  49 .TP
  50 Copyrights [copyright]
  36 Copyrights
  51 Check that each source file contains a copyright notice for the current
  52 year\. You don't need to fix this if you, the potential new copyright holder, ch
```

```
  53 .TP
  54 C style [cstyle]
  41 C style
  55 Check that C source files conform to the Illumos C style rules

  56 .TP
  57 Header check [hdrchk]
  45 Header check
  58 Check that C header files conform to the Illumos header style rules (in addition

  59 .TP
  60 Java style [jstyle]
  49 Java style
  61 Check that Java source files conform to the Illumos Java style rules (which diff

  62 .TP
  63 SCCS Keywords [keywords]
  53 SCCS Keywords
  64 Check that no source files contain unexpanded SCCS keywords\. It is possible tha

  65 .IP
  66 This check does not check for expanded SCCS keywords, though the common \'ident\

  67 .TP
  68 Man page check [manlint]
  69 Check for problems with man pages.
  70 .TP
  71 Mapfile check [mapfilechk]
  60 Mapfile check
  72 Check that linker mapfiles contain a comment directing anyone editing to read th

  63 .SH "OPTIONS"

  73 .TP
  74 Whitespace check [wscheck]
  75 Check for whitespace issues such as mixed tabs/spaces in source files.
  66 \fB\-b branch\fR:

  68 .IP
  69 Compare the current workspace to /branch/ for the purposes of generating file an

  71 .IP
  72 If this option is not specified an attempt is made to determine this automatical

  74 .IP
  75 If no branch is specified and none can be determined automatically \fBorigin/mas

  76 .SH "FILES"
  77 Exception lists can be used to exclude certain files from checking, named after
  78 the specific check.
  79 They can be found in \fB$CODEMGR_WS/exception_lists/\fR, or optionally under
  80 \fB$CODEMGR_WS/\.git/\fR, where they must be suffixed \fB.NOT\fR.
  78 \fBgit nits\fR and \fBgit pbchk\fR support NOT files of the form used by Cadmium

  80 .IP "\(bu" 4
  81 \fBcopyright\.NOT\fR: exclude files listed from copyright checking

  83 .IP "\(bu" 4
  84 \fBcstyle\.NOT\fR: exclude files from the C style check

  86 .IP "\(bu" 4
  87 \fBhdrchk\.NOT\fR: exclude files from the C header style check

  89 .IP "\(bu" 4
  90 \fBkeywords\.NOT\fR: exclude files from the SCCS keywords check
```

```
.IP "\(bu" 4
\fBmapfilechk\.NOT\fR: exclude files from the linker mapfile check

.IP "" 0
```

```
**********************************************************
   12279 Thu Sep 13 19:12:29 2018
new/usr/src/tools/scripts/git-pbchk.py
9803 pbchk could use a -c option
9825 pbchk -b option should be -p
**********************************************************
#!@PYTHON@
#
#  This program is free software; you can redistribute it and/or modify
#  it under the terms of the GNU General Public License version 2
#  as published by the Free Software Foundation.
#
#  This program is distributed in the hope that it will be useful,
#  but WITHOUT ANY WARRANTY; without even the implied warranty of
#  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#  GNU General Public License for more details.
#
#  You should have received a copy of the GNU General Public License
#  along with this program; if not, write to the Free Software
#  Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
#

#
# Copyright (c) 2008, 2010, Oracle and/or its affiliates. All rights reserved.
# Copyright 2008, 2012 Richard Lowe
# Copyright 2014 Garrett D'Amore <garrett@damore.org>
# Copyright (c) 2014, Joyent, Inc.
# Copyright (c) 2015, 2016 by Delphix. All rights reserved.
# Copyright 2016 Nexenta Systems, Inc.
# Copyright 2018 Joyent, Inc.
#

import getopt
import os
import re
import subprocess
import sys
import tempfile

from cStringIO import StringIO

#
# Adjust the load path based on our location and the version of python into
# which it is being loaded.  This assumes the normal onbld directory
# structure, where we are in bin/ and the modules are in
# lib/python(version)?/onbld/Scm/.  If that changes so too must this.
#
sys.path.insert(1, os.path.join(os.path.dirname(__file__), "..", "lib",
                                "python%d.%d" % sys.version_info[:2]))

#
# Add the relative path to usr/src/tools to the load path, such that when run
# from the source tree we use the modules also within the source tree.
#
sys.path.insert(2, os.path.join(os.path.dirname(__file__), ".."))

from onbld.Scm import Ignore
from onbld.Checks import Comments, Copyright, CStyle, HdrChk, WsCheck
from onbld.Checks import JStyle, Keywords, ManLint, Mapfile, SpellCheck


class GitError(Exception):
    pass

def git(command):
    """Run a command and return a stream containing its stdout (and write its
```

```
    stderr to its stdout)"""

    if type(command) != list:
        command = command.split()

    command = ["git"] + command

    try:
        tmpfile = tempfile.TemporaryFile(prefix="git-nits")
    except EnvironmentError, e:
        raise GitError("Could not create temporary file: %s\n" % e)

    try:
        p = subprocess.Popen(command,
                             stdout=tmpfile,
                             stderr=subprocess.PIPE)
    except OSError, e:
        raise GitError("could not execute %s: %s\n" % (command, e))

    err = p.wait()
    if err != 0:
        raise GitError(p.stderr.read())

    tmpfile.seek(0)
    return tmpfile


def git_root():
    """Return the root of the current git workspace"""

    p = git('rev-parse --git-dir')

    if not p:
        sys.stderr.write("Failed finding git workspace\n")
        sys.exit(err)

    return os.path.abspath(os.path.join(p.readlines()[0],
                                        os.path.pardir))


def git_branch():
    """Return the current git branch"""

    p = git('branch')

    if not p:
        sys.stderr.write("Failed finding git branch\n")
        sys.exit(err)

    for elt in p:
        if elt[0] == '*':
            if elt.endswith('(no branch)'):
                return None
            return elt.split()[1]


def git_parent_branch(branch):
    """Return the parent of the current git branch.

    If this branch tracks a remote branch, return the remote branch which is
    tracked.  If not, default to origin/master."""

    if not branch:
        return None

    p = git(["for-each-ref", "--format=%(refname:short) %(upstream:short)",
            "refs/heads/"])

    if not p:
        sys.stderr.write("Failed finding git parent branch\n")
        sys.exit(err)

    for line in p:
        # Git 1.7 will leave a ' ' trailing any non-tracking branch
        if ' ' in line and not line.endswith(' \n'):
            local, remote = line.split()
            if local == branch:
                return remote
    return 'origin/master'


def git_comments(parent):
    """Return a list of any checkin comments on this git branch"""

    p = git('log --pretty=tformat:%%B:SEP: %s..' % parent)

    if not p:
        sys.stderr.write("Failed getting git comments\n")
        sys.exit(err)

    return [x.strip() for x in p.readlines() if x != ':SEP:\n']


def git_file_list(parent, paths=None):
    """Return the set of files which have ever changed on this branch.

    NB: This includes files which no longer exist, or no longer actually
    differ."""

    p = git("log --name-only --pretty=format: %s.. %s" %
             (parent, ' '.join(paths)))

    if not p:
        sys.stderr.write("Failed building file-list from git\n")
        sys.exit(err)

    ret = set()
    for fname in p:
        if fname and not fname.isspace() and fname not in ret:
            ret.add(fname.strip())

    return ret


def not_check(root, cmd):
    """Return a function which returns True if a file given as an argument
    should be excluded from the check named by 'cmd'"""

    ignorefiles = filter(os.path.exists,
                         [os.path.join(root, ".git", "%s.NOT" % cmd),
                          os.path.join(root, "exception_lists", cmd)])
    return Ignore.ignore(root, ignorefiles)


def gen_files(root, parent, paths, exclude):
    """Return a function producing file names, relative to the current
    directory, of any file changed on this branch (limited to 'paths' if
    requested), and excluding files for which exclude returns a true value """

    # Taken entirely from Python 2.6's os.path.relpath which we would use if we
    # could.
    def relpath(path, here):
```

```
        c = os.path.abspath(os.path.join(root, path)).split(os.path.sep)
        s = os.path.abspath(here).split(os.path.sep)
        l = len(os.path.commonprefix((s, c)))
        return os.path.join(*[os.path.pardir] * (len(s)-l) + c[l:])

    def ret(select=None):
        if not select:
            select = lambda x: True

        for f in git_file_list(parent, paths):
            f = relpath(f, '.')
            try:
                res = git("diff %s HEAD %s" % (parent, f))
            except GitError, e:
                # This ignores all the errors that can be thrown. Usually, this
                # that git returned non-zero because the file doesn't exist, but
                # could also fail if git can't create a new file or it can't be
                # executed.  Such errors are 1) unlikely, and 2) will be caught
                # invocations of git().
                continue
            empty = not res.readline()
            if (os.path.isfile(f) and not empty and select(f) and not exclude(f)
                yield f
    return ret


def comchk(root, parent, flist, output):
    output.write("Comments:\n")

    return Comments.comchk(git_comments(parent), check_db=True,
                           output=output)


def mapfilechk(root, parent, flist, output):
    ret = 0

    # We are interested in examining any file that has the following
    # in its final path segment:
    #    - Contains the word 'mapfile'
    #    - Begins with 'map.'
    #    - Ends with '.map'
    # We don't want to match unless these things occur in final path segment
    # because directory names with these strings don't indicate a mapfile.
    # We also ignore files with suffixes that tell us that the files
    # are not mapfiles.
    MapfileRE = re.compile(r'.*((mapfile[^/]*)|(/map\.+[^/]*)|(\.map))$',
        re.IGNORECASE)
    NotMapSuffixRE = re.compile(r'.*\.[ch]$', re.IGNORECASE)

    output.write("Mapfile comments:\n")

    for f in flist(lambda x: MapfileRE.match(x) and not
                   NotMapSuffixRE.match(x)):
        fh = open(f, 'r')
        ret |= Mapfile.mapfilechk(fh, output=output)
        fh.close()
    return ret


def copyright(root, parent, flist, output):
    ret = 0
    output.write("Copyrights:\n")
    for f in flist():
        fh = open(f, 'r')
        ret |= Copyright.copyright(fh, output=output)
        fh.close()
    return ret


def hdrchk(root, parent, flist, output):
    ret = 0
    output.write("Header format:\n")
    for f in flist(lambda x: x.endswith('.h')):
        fh = open(f, 'r')
        ret |= HdrChk.hdrchk(fh, lenient=True, output=output)
        fh.close()
    return ret


def cstyle(root, parent, flist, output):
    ret = 0
    output.write("C style:\n")
    for f in flist(lambda x: x.endswith('.c') or x.endswith('.h')):
        fh = open(f, 'r')
        ret |= CStyle.cstyle(fh, output=output, picky=True,
                             check_posix_types=True,
                             check_continuation=True)
        fh.close()
    return ret


def jstyle(root, parent, flist, output):
    ret = 0
    output.write("Java style:\n")
    for f in flist(lambda x: x.endswith('.java')):
        fh = open(f, 'r')
        ret |= JStyle.jstyle(fh, output=output, picky=True)
        fh.close()
    return ret


def manlint(root, parent, flist, output):
    ret = 0
    output.write("Man page format/spelling:\n")
    ManfileRE = re.compile(r'.*\.[0-9][a-z]*$', re.IGNORECASE)
    for f in flist(lambda x: ManfileRE.match(x)):
        fh = open(f, 'r')
        ret |= ManLint.manlint(fh, output=output, picky=True)
        ret |= SpellCheck.spellcheck(fh, output=output)
        fh.close()
    return ret

def keywords(root, parent, flist, output):
    ret = 0
    output.write("SCCS Keywords:\n")
    for f in flist():
        fh = open(f, 'r')
        ret |= Keywords.keywords(fh, output=output)
        fh.close()
    return ret

def wscheck(root, parent, flist, output):
    ret = 0
    output.write("white space nits:\n")
    for f in flist():
        fh = open(f, 'r')
        ret |= WsCheck.wscheck(fh, output=output)
        fh.close()
    return ret

def run_checks(root, parent, cmds, paths='', opts={}):
    """Run the checks given in 'cmds', expected to have well-known signatures,
```

```
    and report results for any which fail.

    Return failure if any of them did.

    NB: the function name of the commands passed in is used to name the NOT
    file which excepts files from them."""

    ret = 0

    for cmd in cmds:
        s = StringIO()

        exclude = not_check(root, cmd.func_name)
        result = cmd(root, parent, gen_files(root, parent, paths, exclude),
                     output=s)
        ret |= result

        if result != 0:
            print s.getvalue()

    return ret


def nits(root, parent, paths):
    cmds = [copyright,
            cstyle,
            hdrchk,
            jstyle,
            keywords,
            manlint,
            mapfilechk,
            wscheck]
    run_checks(root, parent, cmds, paths)


def pbchk(root, parent, paths):
    cmds = [comchk,
            copyright,
            cstyle,
            hdrchk,
            jstyle,
            keywords,
            manlint,
            mapfilechk,
            wscheck]
    run_checks(root, parent, cmds)


def main(cmd, args):
    parent_branch = None
    checkname = None

    try:
        opts, args = getopt.getopt(args, 'c:p:')
        opts, args = getopt.getopt(args, 'b:')
    except getopt.GetoptError, e:
        sys.stderr.write(str(e) + '\n')
        sys.stderr.write("Usage: %s [-c check] [-p branch] [path...]\n" % cmd)
        sys.stderr.write("Usage: %s [-b branch] [path...]\n" % cmd)
        sys.exit(1)

    for opt, arg in opts:
        # backwards compatibility
        if opt == '-b':
            parent_branch = arg
        elif opt == '-c':
            checkname = arg
        elif opt == '-p':
            parent_branch = arg

    if not parent_branch:
        parent_branch = git_parent_branch(git_branch())

    if checkname is None:
    func = nits
        if cmd == 'git-pbchk':
            checkname= 'pbchk'
        else:
            checkname = 'nits'

    if checkname == 'pbchk':
        func = pbchk
        if args:
            sys.stderr.write("only complete workspaces may be pbchk'd\n");
            sys.exit(1)
        pbchk(git_root(), parent_branch, None)
    elif checkname == 'nits':
        nits(git_root(), parent_branch, args)
    else:
        run_checks(git_root(), parent_branch, [eval(checkname)], args)

    func(git_root(), parent_branch, args)

if __name__ == '__main__':
    try:
        main(os.path.basename(sys.argv[0]), sys.argv[1:])
    except GitError, e:
        sys.stderr.write("failed to run git:\n %s\n" % str(e))
        sys.exit(1)
```