File created: 12-Jun-2024 23:02:26 **{DSK}<home>matt>Interlisp>medley>lispusers>GITFNS.;6**

edit by: mth

changes to: (FNS PRC-COMMAND GIT-BRANCH-RELATIONS GIT-BRANCHES GIT-BRANCH-MENU GIT-PULL-REQUESTS GIT-PRC-BRANCHES CDGITDIR GIT-COMMAND GITORIGIN GIT-RESULT-TO-LINES STRIPLOCAL GIT-WHICH-BRANCH GIT-GET-DIFFERENT-FILES GIT-REMOTE-UPDATE GIT-CHECKOUT GIT-MAKE-BRANCH GIT-MY-BRANCHP GIT-BRANCHES-COMPARE-DIRECTORIES GIT-WORKING-COMPARE-DIRECTORIES)

previous date: 10-Jun-2024 18:43:43 {DSK}<home>matt>Interlisp>medley>lispusers>GITFNS.;5

Read Table: INTERLISP

Package: INTERLISP

Format: XCCS

```
(RPAQQ GITFNSCOMS
       (;; Set up
        (FILES (SYSLOAD FROM LISPUSERS)
               COMPAREDIRECTORIES COMPARESOURCES COMPARETEXT PSEUDOHOSTS JSON UNIXUTILS REGIONMANAGER)
        ;;
        ;; GIT projects
        (COMS (FNS GIT-CLONEP GIT-INIT GIT-MAKE-PROJECT GIT-GET-PROJECT GIT-PUT-PROJECT-FIELD GIT-PROJECT-PATH
                   FIND-ANCESTOR-DIRECTORY GIT-FIND-CLONE GIT-MAINBRANCH GIT-MAINBRANCH?)
              (DECLARE%: EVAL@COMPILE DONTCOPY (RECORDS GIT-PROJECT PULLREQUEST))
              (INITVARS (GIT-DEFAULT-PROJECT 'MEDLEY)
                     [GIT-DEFAULT-PROJECTS '((MEDLEY NIL NIL (EXPORTS.ALL RDSYS RDSYS.LCOM loadups/ patches/
                                                              tmp/ fontsold/ clos/ cltl2/)
                                                    (greetfiles scripts sources library lispusers internal
                                                          doctools rooms))
                                           (NOTECARDS)
                                           (LOOPS)
                                           (TEST)
                                           (MAIKO]
                     (GIT-PROJECTS NIL)
                     (GIT-PRC-MENUS NIL)))
        (P (GIT-INIT))
        (ADDVARS (AROUNDEXITFNS GIT-INIT))
        ;;
        ;; Lisp exec commands
        (INITVARS (GIT-MERGE-COMPARES T)
               (GIT-CDBROWSER-SEPARATE-DIRECTIONS T))
        (COMMANDS gwc bbc prc cob b? cdg cdw)
        (FNS PRC-COMMAND)
        ;;
        ;; File correspondents
        (FNS ALLSUBDIRS MEDLEYSUBDIRS GITSUBDIRS)
        (FNS TOGIT FROMGIT GIT-DELETE-FILE MYMEDLEY-DELETE-FILES)
        (FNS MYMEDLEYSUBDIR GITSUBDIR STRIPDIR STRIPHOST STRIPNAME STRIPWHERE)
        (FNS GFILE4MFILE MFILE4GFILE GIT-REPO-FILENAME)
        ;;
        ;; Git commands
        (FNS GIT-COMMIT GIT-PUSH GIT-PULL GIT-APPROVAL GIT-GET-FILE GIT-FILE-EXISTS? GIT-REMOTE-UPDATE
             GIT-REMOTE-ADD GIT-FILE-DATE GIT-FILE-HISTORY GIT-PRINT-FILE-HISTORY GIT-FETCH)
        ;; Differences
        (FNS GIT-BRANCH-DIFF GIT-COMMIT-DIFFS GIT-BRANCH-RELATIONS)
        ;;
        ;; Branches
        (FNS GIT-BRANCH-NUM GIT-CHECKOUT GIT-WHICH-BRANCH GIT-MAKE-BRANCH GIT-BRANCHES GIT-BRANCH-EXISTS?
             GIT-PICK-BRANCH GIT-BRANCH-MENU GIT-BRANCH-WHENSELECTEDFN GIT-PULL-REQUESTS GIT-SHORT-BRANCH-NAME
             GIT-LONG-NAME GIT-PRC-BRANCHES)
        ;; My branches
        (FNS GIT-MY-CURRENT-BRANCH GIT-MY-BRANCHP GIT-MY-NEXT-BRANCH GIT-MY-BRANCHES)
        ;;
        ;; Worktrees
        (FNS GIT-ADD-WORKTREE GIT-REMOVE-WORKTREE GIT-LIST-WORKTREES WORKTREEDIR)
        ;;
        ;; Comparisons
        (FNS GIT-GET-DIFFERENT-FILES GIT-BRANCHES-COMPARE-DIRECTORIES GIT-WORKING-COMPARE-DIRECTORIES
             GIT-COMPARE-WORKTREE GITCDOBJBUTTONFN GIT-CD-LABELFN GIT-CD-MENUFN GIT-WORKING-COMPARE-FILES
             GIT-BRANCHES-COMPARE-FILES GIT-PR-COMPARE)
        (INITVARS (FROMGITN 0))
        ;;
```

```
        ;; Utilities
        (FNS CDGITDIR GIT-COMMAND GITORIGIN GIT-INITIALS GIT-COMMAND-TO-FILE GIT-RESULT-TO-LINES STRIPLOCAL)
        (PROPS (GITFNS FILETYPE))))

;; Set up

(FILESLOAD (SYSLOAD FROM LISPUSERS)
       COMPAREDIRECTORIES COMPARESOURCES COMPARETEXT PSEUDOHOSTS JSON UNIXUTILS REGIONMANAGER)

;;
;; GIT projects

(DEFINEQ
```

(GIT-CLONEP

```
  [LAMBDA (HOST/DIR NOERROR CHECKANCESTORS)                    ; Edited  1-Oct-2023 18:09 by rmk
                                                               ; Edited 12-May-2022 11:44 by rmk
                                                               ; Edited  8-May-2022 16:24 by rmk
    ;; If CHECKANCESTORS, looks back up the directory chain to see if perhaps the .git is somewhere higher up.
    (IF [AND HOST/DIR (LET [(D (SLASHIT (TRUEFILENAME (PACKFILENAME.STRING 'BODY HOST/DIR 'HOST 'DSK]
                                       (IF (DIRECTORYNAMEP (CONCAT D "/.git/"))
                                           THEN D
                                         ELSEIF (AND CHECKANCESTORS (FIND-ANCESTOR-DIRECTORY D (FUNCTION (LAMBDA (A)
                                                                                                       (DIRECTORYNAMEP
                                                                                                        (CONCAT A
                                                                                                                ".git/"]
      ELSEIF NOERROR
        THEN NIL
      ELSE (ERROR "NOT A GIT CLONE" HOST/DIR])
```

(GIT-INIT

```
  [LAMBDA (EVENT)                                              ; Edited  1-Feb-2023 16:22 by rmk
                                                               ; Edited  1-Oct-2022 12:13 by FGH
                                                               ; Edited  8-Aug-2022 21:52 by lmm
    (SELECTQ EVENT
         ((NIL AFTERMAKESYS AFTERSYSOUT)
              (SETQ GIT-PROJECTS NIL)
              (for X in GIT-DEFAULT-PROJECTS do (APPLY (FUNCTION GIT-MAKE-PROJECT)
                                                       (MKLIST X)))
              NIL)
         NIL])
```

(GIT-MAKE-PROJECT

```
  [LAMBDA (PROJECTNAME CLONEPATH WORKINGPATH EXCLUSIONS DEFAULTSUBDIRS)
                                                               ; Edited  1-Oct-2023 19:33 by rmk
                                                               ; Edited 30-Mar-2023 09:06 by rmk
                                                               ; Edited  5-Feb-2023 12:43 by rmk
                                                               ; Edited  1-Feb-2023 16:55 by rmk
                                                               ; Edited 11-Aug-2022 17:54 by rmk
                                                               ; Edited  9-May-2022 16:20 by rmk
    ;; CLONEPATH must resolve to a git clone.
    ;;    (UNIX-GETENV PROJECTNAME)     Unix variable ROOMS is the full path name.
    ;;    (MEDLEYDIR PROJECTNAME)       e.g. {dsk}/Users/kaplan/medley3.5/loops/
    ;;    (MEDLEYDIR (CONCAT "git-" PROJECTNAME)    e.g. {dsk}/Users/kaplan/medley3.5/git-medley/
    ;;    (MEDLEYDIR (CONCAT PROJECTNAME "DIR")     e.g. {dsk}/Users/kaplan/medley3.5/notecardsdir/
    ;;    (MEDLEYDIR (CONCAT "git-" PROJECTNAME)
    ;; The clone pseudohost is PROJECTNAME  e.g. {NOTECARDS}
    ;; If there is a >working-PROJECTNAME> parallel to clonepath, its pseudhost is WPROJECTNAME, e.g. WNOTECARDS
    ;; Error if clone is not found.
    ;; WORKINGPATH T or NIL means try to find a parallel to the projectpath, T means don't cause an error if not found.
    [SETQ CLONEPATH (if (MEMB CLONEPATH '(NIL T))
                        then ;; The "DIR" handles MEDLEY -> MEDLEYDIR or LOOPS -> LOOPSDIR.
                             ;;
                             (OR (GIT-CLONEP (UNIX-GETENV PROJECTNAME)
                                             T)
                                 (GIT-CLONEP (UNIX-GETENV (PACK* PROJECTNAME "DIR"))
                                             T)
                                 (GIT-CLONEP (MEDLEYDIR (L-CASE PROJECTNAME)
                                                        NIL NIL T)
                                             T)
                                 (GIT-CLONEP (MEDLEYDIR (CONCAT "../" (L-CASE PROJECTNAME))
                                                        NIL NIL T)
                                             T)
                                 (GIT-CLONEP (DIRECTORYNAME (CONCAT MEDLEYDIR "../git-" (L-CASE PROJECTNAME)
                                                                    "/"))
```

```
                                    T)
                                (CL:IF CLONEPATH
                                    (ERROR (CONCAT "Can't find a clone directory for " PROJECTNAME))
                                    (PRINTOUT T "Note: Can't find a clone directory for " PROJECTNAME T)))
                        elseif (GIT-CLONEP [SLASHIT (PACKFILENAME 'HOST 'DSK 'DIRECTORY (UNPACKFILENAME.STRING
                                                                                          (TRUEFILENAME CLONEPATH)
                                                                                          'DIRECTORY
                                                                                          'RETURN]
                                     T T)
                        else (ERROR (CONCAT "Can't find the clone directory " CLONEPATH " for " PROJECTNAME]
    (CL:WHEN CLONEPATH
        (LET (GITIGNORE PROJECT WP)
             (CL:WHEN (SETQ GITIGNORE (INFILEP (PACKFILENAME.STRING 'NAME ".gitignore" 'BODY CLONEPATH)))
                 (SETQ GITIGNORE (CL:WITH-OPEN-FILE (STREAM GITIGNORE)
                                        (bind L until (EOFP STREAM) while (SETQ L (CL:READ-LINE STREAM :EOF-ERROR-P
                                                                                                 NIL :EOF-VALUE NIL))
                                           unless (OR (EQ 0 (NCHARS L))
                                                      (STRPOS "#" L))
                                           collect L))))
             (SETQ EXCLUSIONS (CL:REMOVE-DUPLICATES (APPEND (for E inside EXCLUSIONS collect (MKSTRING E))
                                                            GITIGNORE
                                                            '("deleted/" "*.sysout"))
                                    :TEST
                                    (FUNCTION STRING.EQUAL)))
             ;; We now have the clonepath and the extra parameters for the project.  Do we have a separate working environment?
             (SETQ WP
              (SELECTQ WORKINGPATH
                   ((T NIL)
                        (DIRECTORYNAME (PACKFILENAME.STRING 'HOST 'DSK 'BODY
                                           (CONCAT (SUBSTRING CLONEPATH 1
                                                              (STRPOS "/" CLONEPATH -2 NIL NIL NIL
                                                                      FILEDIRCASEARRAY T))
                                                   "working-"
                                                   (OR (SUBSTRING CLONEPATH
                                                                  (OR (STRPOS CLONEPATH CLONEPATH 1 NIL NIL T
                                                                              FILEDIRCASEARRAY)
                                                                      -2))
                                                       (L-CASE PROJECTNAME))
                                                   ">"))
                                   T))
                   (DIRECTORYNAME (TRUEFILENAME WORKINGPATH)
                          T)))
             [SETQ WORKINGPATH (if WP
                                   then (UNSLASHIT WP)
                                  elseif WORKINGPATH
                                   then (ERROR (CONCAT "Can't find the working directory "
                                                       (AND (EQ WORKINGPATH T)
                                                            "")
                                                       " for " PROJECTNAME]
             (SETQ PROJECT (create GIT-PROJECT
                                   PROJECTNAME _ PROJECTNAME
                                   GITHOST _ (PACK* "{" (PSEUDOHOST PROJECTNAME CLONEPATH)
                                                    "}")
                                   WHOST _ (AND WORKINGPATH (PACK* "{" (PSEUDOHOST (CONCAT "W" PROJECTNAME)
                                                                                   WORKINGPATH)
                                                                   "}"))
                                   EXCLUSIONS _ EXCLUSIONS
                                   DEFAULTSUBDIRS _ (MKLIST DEFAULTSUBDIRS)
                                   CLONEPATH _ CLONEPATH))
             (/RPLACD [OR (ASSOC PROJECTNAME GIT-PROJECTS)
                          (CAR (push GIT-PROJECTS (CONS PROJECTNAME]
                    PROJECT)
             PROJECTNAME))])
```

(GIT-GET-PROJECT

```
  [LAMBDA (PROJECT FIELD NOERROR)                                   ; Edited  7-Jul-2022 11:25 by rmk
                                                                    ; Edited 13-May-2022 10:40 by rmk
                                                                    ; Edited  9-May-2022 20:02 by rmk
                                                                    ; Edited  8-May-2022 11:38 by rmk
    (CL:WHEN (SETQ PROJECT (if (type? GIT-PROJECT PROJECT)
                               then PROJECT
                              elseif (CDR (ASSOC (OR (U-CASE PROJECT)
                                                     GIT-DEFAULT-PROJECT)
                                                 GIT-PROJECTS))
                              elseif NOERROR
                               then NIL
                              else (ERROR "NOT A GIT-PROJECT" PROJECT)))
        (SELECTQ FIELD
             (PROJECTNAME (fetch PROJECTNAME of PROJECT))
             (WHOST (fetch WHOST of PROJECT))
             (GITHOST (fetch GITHOST of PROJECT))
             (EXCLUSIONS (fetch EXCLUSIONS of PROJECT))
             (DEFAULTSUBDIRS
                  (fetch DEFAULTSUBDIRS of PROJECT))
             (CLONEPATH (fetch CLONEPATH of PROJECT))
```

```
            (MAINBRANCH [OR (fetch MAINBRANCH of PROJECT)
                            (replace MAINBRANCH of PROJECT with (OR (GIT-BRANCH-EXISTS? 'origin/main T PROJECT)
                                                                    (GIT-BRANCH-EXISTS? 'origin/master NIL PROJECT
                                                                           ])
           PROJECT))])
```

(GIT-PUT-PROJECT-FIELD

```
  [LAMBDA (PROJECT FIELD NEWVALUE)                                  ; Edited 10-Jun-2023 21:48 by rmk
                                                                    ; Edited 11-Mar-2023 23:00 by rmk
                                                                    ; Edited  7-Jul-2022 11:25 by rmk
                                                                    ; Edited 13-May-2022 10:40 by rmk
                                                                    ; Edited  9-May-2022 20:02 by rmk
                                                                    ; Edited  8-May-2022 11:38 by rmk
    ;; Replaces the value of a project field with NEWVALUE.  The project record is DONTCOPY, to avoid potential name conflicts, so this provides a
    ;; functional interface.  One use:  augment EXCLUSIONS with a list of temporary debug and testing files that you don't want to see in the various
    ;; file listings

    (SETQ PROJECT (GIT-GET-PROJECT PROJECT))
    (SELECTQ FIELD
         (PROJECTNAME (REPLACE PROJECTNAME OF PROJECT WITH NEWVALUE))
         (WHOST (REPLACE WHOST OF PROJECT WITH NEWVALUE))
         (GITHOST (REPLACE GITHOST OF PROJECT WITH NEWVALUE))
         (EXCLUSIONS (REPLACE EXCLUSIONS OF PROJECT WITH NEWVALUE))
         (DEFAULTSUBDIRS
              (REPLACE DEFAULTSUBDIRS OF PROJECT WITH NEWVALUE))
         (CLONEPATH (REPLACE CLONEPATH OF PROJECT WITH NEWVALUE))
         (MAINBRANCH (REPLACE MAINBRANCH OF PROJECT WITH NEWVALUE))
        PROJECT])
```

(GIT-PROJECT-PATH

```
  [LAMBDA (PROJECTNAME PROJECTPATH)                                 ; Edited  8-May-2022 15:10 by rmk
    ;; A project path must identify a clone.  But it may be that a working path (with the convention "my-" is given instead of a "git-".  So, this does a my-
    ;; to git- string substitution, so that we can try again.  Essentially a string-subst of /git-xxx/ for /my-xxx/

    (SETQ PROJECTPATH (TRUEFILENAME PROJECTPATH))
    (CL:UNLESS (MEMB (NTHCHARCODE PROJECTPATH -1)
                     (CHARCODE (> /)))
        (SETQ PROJECTPATH (CONCAT PROJECTPATH "/")))
    (LET (MY-POS (MYSUBDIR (CONCAT "/my-" PROJECTNAME "/")))
         (CL:WHEN (SETQ MY-POS (STRPOS MYSUBDIR PROJECTPATH 1 NIL NIL NIL FILEDIRCASEARRAY))
             (SLASHIT [CONCAT (SUBSTRING PROJECTPATH 1 MY-POS)
                              "git-" PROJECTNAME (SUBSTRING PROJECTPATH (IPLUS -1 MY-POS (NCHARS MYSUBDIR]
                    T))])
```

(FIND-ANCESTOR-DIRECTORY

```
  [LAMBDA (STARTDIR PREDFN)                                         ; Edited  8-May-2022 12:17 by rmk
    (BIND POS (A _ STARTDIR) WHILE (SETQ POS (STRPOS "/" A -2 NIL NIL NIL FILEDIRCASEARRAY T))
       DO (SETQ A (SUBSTRING A 1 POS))
          (CL:WHEN (APPLY* PREDFN A)
                 (RETURN A])
```

(GIT-FIND-CLONE

```
  [LAMBDA (PROJECTNAME PROJECTPATH)                                 ; Edited  8-May-2022 15:00 by rmk
    ;; Maybe the PROJECTPATH was actually a MY path, in which case our best guess is that the git-clone is a sister somewhere above.

    (OR (GIT-CLONEP PROJECTPATH T T)
        (GIT-CLONEP (GIT-PROJECT-PATH PROJECTNAME PROJECTPATH)
               T T)
        [FIND-ANCESTOR-DIRECTORY PROJECTPATH (FUNCTION (LAMBDA (A)
                                                         (BIND D (GEN _ (\GENERATEFILES A NIL NIL 1))
                                                            WHILE (SETQ D (\GENERATENEXTFILE GEN))
                                                            WHEN (GIT-CLONEP D T)
                                                            DO (RETFROM (FUNCTION FIND-ANCESTOR-DIRECTORY)
                                                                        D]
        (ERROR "NOT A GIT CLONE" PROJECTPATH])
```

(GIT-MAINBRANCH

```
  [LAMBDA (PROJECT LOCAL NOERROR)                                   ; Edited  7-Jul-2022 11:16 by rmk
                                                                    ; Edited  9-May-2022 16:34 by rmk

    (LET ((MB (GIT-GET-PROJECT PROJECT 'MAINBRANCH NOERROR)))
         (CL:IF LOCAL
             (CONCAT "local/" (STRIPWHERE MB))
             MB)])
```

(GIT-MAINBRANCH?

```
  [LAMBDA (BRANCH PROJECT NOERROR)                                  ; Edited  9-Aug-2022 10:40 by rmk
                                                                    ; Edited  9-May-2022 15:06 by rmk
    (IF (STRING.EQUAL (STRIPWHERE (GIT-MAINBRANCH PROJECT NIL T)
                                  T)
              (STRIPWHERE BRANCH))
      ELSEIF NOERROR
```

```
       THEN NIL
      ELSE (ERROR "Can't modify main branch" BRANCH])

)

(DECLARE%: EVAL@COMPILE DONTCOPY

(DECLARE%: EVAL@COMPILE

(TYPERECORD GIT-PROJECT (PROJECTNAME GITHOST WHOST EXCLUSIONS DEFAULTSUBDIRS CLONEPATH MAINBRANCH))

(RECORD PULLREQUEST (PRNUMBER PRDESCRIPTION PRNAME PRSTATUS PRPROJECT PRURL PRLOGIN))
)
)

(RPAQ? GIT-DEFAULT-PROJECT 'MEDLEY)

(RPAQ? GIT-DEFAULT-PROJECTS
       '((MEDLEY NIL NIL (EXPORTS.ALL RDSYS RDSYS.LCOM loadups/ patches/ tmp/ fontsold/ clos/ clt12/)
                (greetfiles scripts sources library lispusers internal doctools rooms))
         (NOTECARDS)
         (LOOPS)
         (TEST)
         (MAIKO)))

(RPAQ? GIT-PROJECTS NIL)

(RPAQ? GIT-PRC-MENUS NIL)

(GIT-INIT)

(ADDTOVAR AROUNDEXITFNS GIT-INIT)

;;
;; Lisp exec commands

(RPAQ? GIT-MERGE-COMPARES T)

(RPAQ? GIT-CDBROWSER-SEPARATE-DIRECTIONS T)


(DEFCOMMAND gwc (SUBDIR . OTHERS)
    ;; Compares the specified local git-medley subdirectories against my working Medley.  The SUBDIRS are the arguments up to one that looks like a
    ;; project
     (LET ((SUBDIRS (AND SUBDIR (CONS SUBDIR OTHERS)))
           PROJECT)
          (SETQ SUBDIRS (FOR STAIL ON SUBDIRS COLLECT (IF (GIT-GET-PROJECT (CAR STAIL)
                                                                 NIL T)
                                                             THEN (SETQ PROJECT (CAR STAIL))
                                                                  (GO $$OUT))
                                                         (CAR STAIL)))
          (GIT-WORKING-COMPARE-DIRECTORIES SUBDIRS NIL NIL NIL T PROJECT)))


(DEFCOMMAND bbc (BRANCH1 BRANCH2 LOCAL PROJECT)
    ;; Compares 2 git branches.  Defaults to local/ if LOCAL, otherwise defaults to origin/.  BRANCH2 defaults to the main branch (origin/ or local/
    ;; depending on LOCAL)
     (SETQ PROJECT (GIT-GET-PROJECT PROJECT))
    (GIT-FETCH PROJECT)
    (SETQ BRANCH1 (SELECTQ (U-CASE BRANCH1)
                        ((NIL T)
                             (GIT-MY-CURRENT-BRANCH PROJECT))
                        ((LOCAL REMOTE ORIGIN)
                             (GIT-PICK-BRANCH (GIT-BRANCHES BRANCH1 PROJECT T)))
                        (OR (GIT-LONG-NAME BRANCH1 NIL PROJECT)
                            BRANCH1)))
    (SETQ BRANCH2 (SELECTQ (U-CASE BRANCH2)
                        ((NIL T)
                             (GIT-MAINBRANCH PROJECT LOCAL))
                        ((LOCAL REMOTE ORIGIN)
                             (GIT-PICK-BRANCH (GIT-BRANCHES BRANCH2 PROJECT T)))
                        (OR (GIT-LONG-NAME BRANCH2 NIL PROJECT)
                            BRANCH2)))
    (GIT-BRANCHES-COMPARE-DIRECTORIES BRANCH1 (OR BRANCH2 (GIT-MAINBRANCH PROJECT LOCAL))
           LOCAL PROJECT))


(DEFCOMMAND prc (REMOTEBRANCH DRAFTS PROJECT)
    ;; Compares REMOTEBRANCH against the main orign branch, for pull-request assessment
     (PRC-COMMAND REMOTEBRANCH DRAFTS PROJECT))


(DEFCOMMAND cob (BRANCH NEXTTITLESTRING PROJECT)
```

;; Switches to BRANCH. T means my current branch, NEW/NEXT means my next branch (under wherever we are now), and NEXTTITLESTRING if
;; given will be attached to the branch-name. Default is to bring up a menu of locally available branches.

```
  (CL:UNLESS (STRINGP NEXTTITLESTRING)
         (SETQ PROJECT NEXTTITLESTRING))
 (CL:UNLESS PROJECT
     (CL:WHEN (GIT-GET-PROJECT BRANCH NIL T)
         (SETQ PROJECT BRANCH)
         (SETQ BRANCH NIL)))
 (SETQ PROJECT (GIT-GET-PROJECT PROJECT))
 (GIT-FETCH PROJECT)
 (SELECTQ (U-CASE BRANCH)
      (T (GIT-CHECKOUT (GIT-MY-CURRENT-BRANCH PROJECT)
                PROJECT))
      ((NEW NEXT)
          (GIT-MAKE-BRANCH NIL NEXTTITLESTRING PROJECT))
      (CL:WHEN [SETQ BRANCH (IF BRANCH
                                THEN (GIT-LONG-NAME BRANCH NIL PROJECT)
                               ELSE (GIT-PICK-BRANCH (GIT-BRANCHES 'LOCAL PROJECT T)
                                                     (CONCAT (L-CASE (GIT-GET-PROJECT PROJECT 'PROJECTNAME)
                                                                     T)
                                                             " branches"]
           (GIT-CHECKOUT BRANCH PROJECT))))


(DEFCOMMAND b? (PROJECT) (SETQ PROJECT (GIT-GET-PROJECT PROJECT))
   (GIT-FETCH PROJECT)
   (CONCAT (L-CASE (GIT-GET-PROJECT PROJECT 'PROJECTNAME)
                   T)
           " "
           (GIT-WHICH-BRANCH PROJECT)))


(DEFCOMMAND cdg (PROJECT SUBDIR) (CL:UNLESS (GIT-GET-PROJECT PROJECT NIL T)
                                        (SETQ SUBDIR PROJECT)
                                        (SETQ PROJECT GIT-DEFAULT-PROJECT))
   (CL:WHEN [AND SUBDIR (NOT (MEMB (CHCON1 SUBDIR))
                            (CHARCODE (> /]
       (SETQ SUBDIR (CONCAT SUBDIR "/")))
   (SLASHIT (/CNDIR (CONCAT (GIT-GET-PROJECT PROJECT 'GITHOST)
                            (OR SUBDIR "")))
           T))


(DEFCOMMAND cdw (PROJECT SUBDIR) (CL:UNLESS (GIT-GET-PROJECT PROJECT NIL T)
                                        (SETQ SUBDIR PROJECT)
                                        (SETQ PROJECT GIT-DEFAULT-PROJECT))
   (CL:WHEN [AND SUBDIR (NOT (MEMB (CHCON1 SUBDIR))
                            (CHARCODE (> /]
       (SETQ SUBDIR (CONCAT SUBDIR "/")))
   (SLASHIT (/CNDIR (CONCAT (GIT-GET-PROJECT PROJECT 'WHOST)
                            (OR SUBDIR "")))
           T))

(DEFINEQ
```

(PRC-COMMAND

```
  [LAMBDA (REMOTEBRANCH DRAFTS PROJECT)                    ; Edited 13-May-2024 18:49 by rmk
                                                           ; Edited  2-May-2024 11:44 by rmk
                                                           ; Edited  1-Apr-2024 20:24 by rmk
                                                           ; Edited 28-Jul-2023 09:03 by rmk
```

;; DRAFTS can be DRAFT(S), NODRAFTS, or NIL. If DRAFTS, then only draft PR's are shown, of NODRAFTS then only nondrafts are shown.
;; Anything else, both drafts and nondrafts are shown in the menu.

```
    (LET (PRS MENUWINDOW OLDMENUWINDOW)
         (if PROJECT
             then (SETQ PROJECT (GIT-GET-PROJECT PROJECT))
           elseif (GIT-GET-PROJECT REMOTEBRANCH NIL T)
             then (SETQ PROJECT REMOTEBRANCH)
                  (SETQ REMOTEBRANCH NIL)
           elseif (GIT-GET-PROJECT DRAFTS NIL T)
             then (SETQ PROJECT DRAFTS)
                  (SETQ DRAFTS NIL))
         (CL:UNLESS PROJECT (SETQ PROJECT GIT-DEFAULT-PROJECT))
         (SELECTQ (U-CASE REMOTEBRANCH)
              ((DRAFT DRAFTS)
                   (SETQ REMOTEBRANCH NIL)
                   (SETQ DRAFTS 'DRAFTS))
              ((NODRAFT NODRAFTS)
                   (SETQ REMOTEBRANCH NIL)
                   (SETQ DRAFTS 'NODRAFTS))
              NIL)
         (GIT-FETCH PROJECT)
         ;; Always include drafts??
         (SETQ PRS (GIT-PULL-REQUESTS (NEQ 'NODRAFTS DRAFTS)
                                      PROJECT))
```

```
        ;; Filter by REMOTEBRANCH properties
        (SETQ PRS (for PR FOUND in PRS when (if (STRING-EQUAL "Interlisp" (fetch PRLOGIN of PR))
                                                then (OR (NULL REMOTEBRANCH)
                                                         (STRPOS REMOTEBRANCH (fetch PRDESCRIPTION of PR)
                                                                 NIL NIL NIL NIL FILEDIRCASEARRAY)
                                                         (STRPOS REMOTEBRANCH (fetch PRNAME of PR)
                                                                 NIL NIL NIL NIL FILEDIRCASEARRAY))
                                              else (CL:UNLESS FOUND
                                                       (SETQ FOUND T)
                                                       (PRINTOUT T "Ignored because not owned by Interlisp: " T))
                                                   (PRINTOUT T 3 (fetch PRDESCRIPTION of PR)
                                                          "  ("
                                                          (fetch PRLOGIN of PR)
                                                          ")" T)
                                                   NIL)
                          collect PR))
        (if PRS
            then (if (CDR PRS)
                     then (SETQ MENUWINDOW (ADDMENU (GIT-BRANCH-MENU (GIT-PRC-BRANCHES DRAFTS PROJECT PRS)
                                                                     (CONCAT (LENGTH PRS)
                                                                             " pull requests"))
                                                    NIL NIL T))
                          ;; Position the new menu just under the current TTY window, to keep it out of the way of the comparison windows.  If
                          ;; we have menus open for other projects, those probably should be pushed down to make room for the new menu,
                          ;; and moved up when a higher menu is closed.  An edge case that is not worth the effort.
                          [MOVEW MENUWINDOW (RELCREATEPOSITION '(TTY 0)
                                                               `(TTY 0 ,(IDIFFERENCE -2 (fetch HEIGHT of (WINDOWREGION
                                                                                                          MENUWINDOW]
                          (CL:WHEN [OPENWP (CDR (SETQ OLDMENUWINDOW (ASSOC PROJECT GIT-PRC-MENUS]
                              (CLOSEW (CDR OLDMENUWINDOW)))
                          (OPENW MENUWINDOW)
                          (RPLACD [OR OLDMENUWINDOW (CAR (push GIT-PRC-MENUS (CONS PROJECT]
                                  MENUWINDOW)
                          MENUWINDOW
                     else (GIT-PR-COMPARE (fetch PRNAME of (CAR PRS))
                                          PROJECT))
            else (CONCAT "No open " (OR REMOTEBRANCH "")
                         " pull requests"])

)

;;
;; File correspondents

(DEFINEQ

(ALLSUBDIRS
  [LAMBDA (PROJECT)
    ;; Edited 13-May-2022 10:40 by rmk
    ;; Edited 10-May-2022 00:16 by rmk
    ;; Edited  7-May-2022 16:58 by rmk: the union of the subdirectories that exist in the project
    (LET ((HOSTS (MKLIST (FETCH GITHOST OF PROJECT)))
          VAL)
         (CL:WHEN (FETCH WHOST OF PROJECT)
             (PUSHNEW HOSTS (FETCH WHOST OF PROJECT)))
         (SORT (FOR H VAL IN HOSTS
                  JOIN (FOR F D IN (FILDIR (PACKFILENAME 'HOST H 'BODY '*)
                                           1)
                          WHEN (DIRECTORYNAMEP F)
                          UNLESS (OR [EQ (CHARCODE %.)
                                         (CHCON1 (SETQ D (FILENAMEFIELD F 'DIRECTORY]
                                     (THEREIS SKIP IN (FETCH EXCLUSIONS OF PROJECT)
                                        FIRST (SETQ D (CONCAT D "/"))
                                        SUCHTHAT (STRPOS SKIP D 1 NIL T NIL FILEDIRCASEARRAY)))
                          DO [SETQ D (UNSLASHIT (L-CASE (SUBSTRING D 1 -2]
                              (CL:UNLESS (MEMBER D VAL)
                                     (PUSH VAL D)))
                  FINALLY (RETURN VAL])

(MEDLEYSUBDIRS
  [LAMBDA (PROJECT ALLSUBDIRS)                                    ; Edited 13-May-2022 10:40 by rmk
                                                                  ; Edited  7-May-2022 23:15 by rmk
    (FOR D IN (OR ALLSUBDIRS (ALLSUBDIRS PROJECT)) COLLECT (UNSLASHIT (PACKFILENAME 'HOST
                                                                                    (FETCH WHOST OF PROJECT)
                                                                                    'DIRECTORY D)
                                                                      T])

(GITSUBDIRS
  [LAMBDA (PROJECT ALLSUBDIRS)                                    ; Edited 10-May-2022 00:23 by rmk
                                                                  ; Edited  7-May-2022 23:14 by rmk
```

```
                                                                    ; Edited  4-Feb-2022 18:06 by rmk
    (FOR D IN (OR ALLSUBDIRS (ALLSUBDIRS PROJECT)) COLLECT (SLASHIT (PACKFILENAME 'HOST (FETCH GITHOST
                                                                                          OF PROJECT)
                                                                                 'DIRECTORY D)
                                                                        T])

)

(DEFINEQ
```

(TOGIT

```
  [LAMBDA (MFILES PROJECT)                                          ; Edited 10-May-2022 10:45 by rmk
                                                                    ; Edited  7-May-2022 23:15 by rmk
```

;; Does anybody call this?

```
    (SETQ PROJECT (GIT-GET-PROJECT PROJECT))
```

;; Copies MFILES to {GIT}. We do a sanity check to make sure particular MFILE is the latest version--we may have created another one without ;; revising the directory browser.

```
    (GIT-MAINBRANCH? (GIT-WHICH-BRANCH PROJECT)
          PROJECT)
    (FOR MF GF DEST (MEDLEYSUBDIRS _ (MEDLEYSUBDIRS PROJECT)) INSIDE MFILES
       COLLECT (SETQ MF (OR (FINDFILE MF NIL MEDLEYSUBDIRS)
                            (ERROR "FILE NOT FOUND" MF)))
               (CL:UNLESS (STRING.EQUAL MF (INFILEP (PACKFILENAME 'VERSION NIL 'BODY MF))
                                FILEDIRCASEARRAY)
                   (FLASHWINDOW T)
                   (PRIN3 (CONCAT MF " is not the latest version!")
                          T)
                   (ERROR!))
               (SETQ GF (GFILE4MFILE MF PROJECT))
               (PRIN3 (IF (SETQ DEST (COPYFILE MF GF))
                          THEN (CONCAT "Copied to " GF)
                        ELSE (FLASHWINDOW T)
                             (CONCAT MF " cannot be copied"))
                      T)
               DEST])
```

(FROMGIT

```
  [LAMBDA (GFILES PROJECT)                                          ; Edited 10-May-2022 10:45 by rmk
                                                                    ; Edited  4-Feb-2022 18:08 by rmk
                                                                    ; Edited 18-Jan-2022 16:31 by rmk
```

;; Does anybody call this?

```
    (SETQ PROJECT (GIT-GET-PROJECT PROJECT))
    (FOR GF MF DEST (GITSUBDIRS _ (GITSUBDIRS PROJECT)) INSIDE GFILES
       COLLECT (SETQ GF (OR (FINDFILE GF NIL GITSUBDIRS)
                            (ERROR "FILE NOT FOUND" GF)))
               (SETQ MF (MFILE4GFILE GF))
               (PRIN3 (IF (SETQ DEST (COPYFILE GF MF))
                          THEN (CONCAT "Copied to " DEST)
                               DEST
                        ELSE (FLASHWINDOW T)
                             (CONCAT GF " cannot be copied"))
                      T)
               DEST])
```

(GIT-DELETE-FILE

```
  [LAMBDA (FILE PROJECT)                                            ; Edited  8-May-2022 09:27 by rmk
                                                                    ; Edited 18-Jan-2022 23:07 by rmk
                                                                    ; Edited 19-Dec-2021 16:11 by rmk
                                                                    ; Edited 16-Dec-2021 13:00 by rmk
```

;; This deletes a file in the local checkout git directory {UNIX}... FILE has to already be a full file name, for safety.

;; Since git files are on UNIX, we don't have to worry about older version numbers.

;; We could make this undoable by copying it to deleted/, but git also can restore.

```
    (GIT-CLONEP FILE NIL T)
    (DELFILE FILE])
```

(MYMEDLEY-DELETE-FILES

```
  [LAMBDA (FILE PROJECT)                                            ; Edited 13-May-2022 10:40 by rmk
                                                                    ; Edited  8-May-2022 23:31 by rmk
```

;; FILE is presumably the latest version of a file in the MyMedley directory, and we are presumably removing all versions of that file. If we left older ;; versions, we would really trash ourselves.

;; But to guard against mistakes, "deletion" consists of moving all versions of the file from its current location to a deleted/ subdirectory of ;; MEDLEYDIR, one that does not correspond to a git subdirectory.

```
    (SETQ FILE (CONTRACT.PH FILE (FETCH WHOST OF PROJECT)))
    (CL:WHEN (EQ (FILENAMEFIELD (FETCH WHOST OF PROJECT)
                        'HOST)
                 (FILENAMEFIELD FILE 'HOST))
        (FOR F IN (DREVERSE (FILDIR (PACKFILENAME 'VERSION '* 'BODY FILE)))
           COLLECT
```

```
                ;; Delete the earlier ones first, if it goes bad, you don't want them to persist
                (CL:UNLESS (RENAMEFILE F (PACKFILENAME 'DIRECTORY (CONCAT "deleted>" (FILENAMEFIELD
                                                                                     F
                                                                                     'DIRECTORY))
                                                                  'BODY F))
                       (ERROR "Could not delete " F))
                F))])

)

(DEFINEQ

(MYMEDLEYSUBDIR
  [LAMBDA (SUBDIR STAR PROJECT)                                    ; Edited 13-May-2022 10:40 by rmk
                                                                   ; Edited  7-May-2022 23:15 by rmk
    (UNSLASHIT (PACK* (PACKFILENAME 'HOST (FETCH WHOST OF PROJECT)
                           'DIRECTORY SUBDIR)
                      (CL:IF STAR
                          "*"
                          "")])

(GITSUBDIR
  [LAMBDA (SUBDIR STAR PROJECT)                                    ; Edited  7-May-2022 20:39 by rmk
                                                                   ; Edited 26-Feb-2022 11:56 by rmk
    (SLASHIT (PACK* (PACKFILENAME 'HOST (FETCH GITHOST OF PROJECT)
                           'DIRECTORY SUBDIR)
                    (CL:IF STAR
                        "*"
                        "")])

(STRIPDIR
  [LAMBDA (FILE DIRECTORY)                                         ; Edited 18-Jan-2022 16:09 by rmk
                                                                   ; Edited  8-Nov-2021 11:50 by rmk:
    (IF (STRPOS DIRECTORY FILE 1 NIL T NIL FILEDIRCASEARRAY)
        THEN (SUBSTRING FILE (ADD1 (NCHARS DIRECTORY)))
      ELSE FILE])

(STRIPHOST
  [LAMBDA (NAME)                                                   ; Edited 18-Jan-2022 15:37 by rmk
    (LET ((POS (STRPOS "}" NAME)))
         (CL:IF POS
             (SUBSTRING NAME (ADD1 POS))
             NAME)])

(STRIPNAME
  [LAMBDA (FILE)
    ;; Edited  5-Feb-2022 08:38 by rmk: the name/ext/version of FILE without disturbing host or directory.  Strips everything after last / >
    ;; Removes the name/ext/version of FILE without disturbing host or directory.  Strips everything after last / >
    (FOR I LASTDIRPOS FROM 1 DO (SELCHARQ (NTHCHARCODE FILE I)
                                          ((> < /)
                                               (SETQ LASTDIRPOS I))
                                          (NIL (RETURN (CL:IF LASTDIRPOS
                                                              (SUBSTRING FILE 1 LASTDIRPOS)
                                                              FILE)))
                                          NIL])

(STRIPWHERE
  [LAMBDA (BRANCH ORIGINTOO)                                       ; Edited  9-Aug-2022 10:39 by rmk
                                                                   ; Edited  4-Aug-2022 10:31 by rmk
                                                                   ; Edited  9-May-2022 14:31 by rmk
    ;; Leave origin/ unless ORIGINTOO
    (LET ((POS (STRPOS "/" BRANCH)))
         (CL:IF [AND POS (MEMB [L-CASE (MKATOM (SUBSTRING BRANCH 1 (SUB1 POS]
                               (CL:IF ORIGINTOO
                                   '(local origin)
                                   '(local))]
             (SUBSTRING BRANCH (ADD1 POS))
             BRANCH)])

)

(DEFINEQ

(GFILE4MFILE
  [LAMBDA (MFILE PROJECT)                                          ; Edited  7-May-2022 23:19 by rmk
                                                                   ; Edited  4-Feb-2022 18:04 by rmk
    (SLASHIT (PACKFILENAME 'HOST (FETCH GITHOST OF PROJECT)
                    'VERSION NIL 'BODY MFILE)
          T])
```

(MFILE4GFILE

```
  [LAMBDA (GFILE PROJECT)                                    ; Edited 13-May-2022 10:40 by rmk
                                                             ; Edited  7-May-2022 23:20 by rmk
                                                             ; Edited  4-Feb-2022 18:04 by rmk
                                                             ; Edited 18-Jan-2022 15:24 by rmk
    (UNSLASHIT (PACKFILENAME 'HOST (FETCH WHOST OF PROJECT)
                      'VERSION NIL 'BODY GFILE])
```

(GIT-REPO-FILENAME

```
  [LAMBDA (GFILE PROJECT)                                    ; Edited  8-May-2022 23:35 by rmk
    ;; Returns the string that the repo expects for a file name.  The prefix is stripped, brackets go to slashes, subdirectories are lower cased, an initial /
    ;; and a final period is remove.
    (SETQ GFILE (SLASHIT [IF (EQ (FILENAMEFIELD (FETCH GITHOST OF PROJECT)
                                                'HOST)
                                 (FILENAMEFIELD GFILE 'HOST))
                            THEN (STRIPHOST GFILE)
                           ELSE (STRIPDIR GFILE (TRUEFILENAME (FETCH GITHOST OF PROJECT]
                         T))
    (CL:WHEN (EQ (CHARCODE /)
                 (CHCON1 GFILE))
        (SETQ GFILE (SUBSTRING GFILE 2)))
    (CL:WHEN (EQ (CHARCODE %.)
                 (NTHCHARCODE GFILE -1))
        (SETQ GFILE (SUBSTRING GFILE 1 -2)))
    GFILE])

)

;;
;; Git commands

(DEFINEQ
```

(GIT-COMMIT

```
  [LAMBDA (FILES TITLE MESSAGE PROJECT)                      ; Edited  9-May-2022 16:11 by rmk
                                                             ; Edited 16-Nov-2021 08:06 by rmk:
                                                             ; Edited  2-Nov-2021 21:26 by rmk:
    ;; Commits files that are already in the (non-main) current git branch.
    (CL:WHEN (STRING.EQUAL (GIT-MAINBRANCH PROJECT)
                           (GIT-WHICH-BRANCH PROJECT))
           (ERROR "Cannot commit to the main branch"))
    (HELP "UNIMPLEMENTED")
    (GIT-MAINBRANCH? (GIT-WHICH-BRANCH PROJECT)
           PROJECT)
    (LET (GFILES)
         (SETQ GFILES (FOR F GF INSIDE FILES COLLECT (SETQ GF (INFILEP (GFILE4MFILE F PROJECT])
```

(GIT-PUSH

```
  [LAMBDA (BRANCH PROJECT)                                   ; Edited  2-May-2024 11:23 by mth
                                                             ; Edited  9-May-2022 15:06 by rmk
                                                             ; Edited  8-Dec-2021 22:32 by rmk
                                                             ; Edited 16-Nov-2021 08:06 by rmk:
                                                             ; Edited  2-Nov-2021 21:34 by rmk:
    (CL:UNLESS BRANCH
        (SETQ BRANCH (GIT-WHICH-BRANCH PROJECT)))
    (GIT-MAINBRANCH? BRANCH PROJECT)
    (GIT-COMMAND (CONCAT "git push %"" BRANCH "%"")
           NIL NIL PROJECT])
```

(GIT-PULL

```
  [LAMBDA (BRANCH PROJECT)                                   ; Edited  2-May-2024 11:22 by mth
                                                             ; Edited  9-May-2022 15:07 by rmk
                                                             ; Edited  8-Dec-2021 22:47 by rmk
                                                             ; Edited 16-Nov-2021 08:06 by rmk:
                                                             ; Edited  2-Nov-2021 21:34 by rmk:
    (SETQ PROJECT (GIT-GET-PROJECT PROJECT))
    (GIT-COMMAND (CONCAT "git pull %"" (OR BRANCH (GIT-WHICH-BRANCH PROJECT))
                         "%"")
           NIL NIL PROJECT])
```

(GIT-APPROVAL

```
  [LAMBDA (BRANCH PROJECT)                                   ; Edited  9-May-2022 15:08 by rmk
                                                             ; Edited 19-Nov-2021 15:08 by rmk:
    (GIT-ADD-WORKTREE BRANCH T PROJECT)
    (GIT-ADD-WORKTREE (GIT-MAINBRANCH PROJECT)
           T])
```

(GIT-GET-FILE

```
[LAMBDA (BRANCH GITFILE LOCALFILE NOERROR PROJECT)
```

;; Edited 2-May-2024 12:08 by mth

;; Edited 18-Jul-2022 09:18 by rmk

;; Edited 8-Jul-2022 10:36 by rmk

;; Edited 5-Jul-2022 00:09 by rmk: Redirect show command to tmp/ rename to localfile

;; Edited 30-Jun-2022 22:09 by rmk

;; Edited 22-May-2022 17:34 by rmk

;; Edited 8-May-2022 16:54 by rmk: the stream, not the name because of the NODIRCORE case.

;; Edited 6-Mar-2022 17:45 by rmk: the stream, not the name because of the NODIRCORE case.

;; Returns the stream, not the name because of the NODIRCORE case.

;; If GITFILE in (remote) BRANCH exists, it is copied to LOCALFILE and LOCALFILE is returned. If it doesn't exist, return value is NIL if ;; NOERROR, otherwise an ERROR.

```
(CL:WHEN (AND BRANCH (STRPOS "local/" BRANCH 1 NIL T))
    (SETQ BRANCH (SUBSTRING BRANCH 7)))
(LET ((RESULTFILE (GIT-COMMAND-TO-FILE (CONCAT "git show %"" BRANCH ":" GITFILE "%"")
                          PROJECT T))
      TYPE DATE)
     (CL:WHEN (LISTP RESULTFILE)                              ; CADR is Unix error stream
         (CL:WITH-OPEN-FILE (ESTREAM (CADR RESULTFILE)
                                     :DIRECTION :INPUT :EXTERNAL-FORMAT (SYSTEM-EXTERNALFORMAT))
                (COPYCHARS ESTREAM T))
         (DELFILE (CADR RESULTFILE))
         (SETQ RESULTFILE (CAR RESULTFILE)))
     (if RESULTFILE
         then (CL:MULTIPLE-VALUE-SETQ (TYPE DATE)
                     (LISPFILETYPE RESULTFILE))
              (CL:WHEN (OR DATE (SETQ DATE (GIT-FILE-DATE GITFILE BRANCH PROJECT NOERROR)))
                  (SETFILEINFO RESULTFILE 'CREATIONDATE DATE))
              (RENAMEFILE RESULTFILE LOCALFILE)
       elseif NOERROR
         then NIL
       else (ERROR "GIT FILE NOT FOUND" GITFILE])
```

(GIT-FILE-EXISTS?

```
[LAMBDA (GFILE BRANCH PROJECT)                                ; Edited 5-Jul-2022 10:27 by rmk
```

;; If the noerror DATE is NIL, the file doesn't exist.

```
(CL:WHEN (GIT-FILE-DATE GFILE BRANCH PROJECT T)
       T])
```

(GIT-REMOTE-UPDATE

```
[LAMBDA (DOIT PROJECT)
  (DECLARE (USEDFREE LAST-REMOTE-UPDATE-IDATE))               ; Edited 12-Jun-2024 12:57 by mth
                                                              ; Edited 7-May-2022 22:41 by rmk
```

;; Because git hangs on this (and other things), do this no more than once a day

```
(CL:WHEN [OR DOIT (NOT (BOUNDP 'LAST-REMOTE-UPDATE-IDATE))
             (IGREATERP (IDIFFERENCE (IDATE)
                                     LAST-REMOTE-UPDATE-IDATE)
                        (CONSTANT (TIMES 24 60 60 1000]
    (PRINTOUT T "Updating from remote, local branch is " (GIT-WHICH-BRANCH PROJECT T)
          T)
    (PROG1 (GIT-COMMAND "git remote update origin" NIL PROJECT)
        (SETQ LAST-REMOTE-UPDATE-IDATE (IDATE)))])
```

(GIT-REMOTE-ADD

```
[LAMBDA (NAME URL)                                            ; Edited 31-Jan-2022 13:53 by rmk
  (LET [(RESULT (GIT-COMMAND (CONCAT "git remote add " NAME " " URL]
```

;; Does it return an error line? What if URL is not good?

```
       (CAR RESULT])
```

(GIT-FILE-DATE

```
[LAMBDA (GFILE BRANCH PROJECT NOERROR)                        ; Edited 2-May-2024 11:22 by mth
                                                              ; Edited 6-Jul-2022 19:39 by rmk
                                                              ; Edited 5-Jul-2022 10:30 by rmk
(CL:WHEN (AND NIL BRANCH (STRPOS "local/" BRANCH 1 NIL T))
    (SETQ BRANCH (SUBSTRING BRANCH 7)))
(LET [(DATE (CAR (GIT-COMMAND (CONCAT "git log -1 --pretty=%"format:%%cD%" " (CL:IF BRANCH
                                                                                    (CONCAT "%"" BRANCH "%" --
")
                                                                                    "")
                                      (GIT-REPO-FILENAME GFILE PROJECT))
                      NIL T PROJECT]
     (CL:UNLESS (OR DATE NOERROR)
```

;; We suppressed the generic error in GIT-COMMAND, so we could do our own thing

```
            (ERROR "GIT FILE NOT FOUND" GFILE))
          DATE])
```

(GIT-FILE-HISTORY

```
  [LAMBDA (FILE PROJECT PRINT)                                    ; Edited  4-Jul-2022 23:09 by rmk
                                                                  ; Edited  1-Jul-2022 22:57 by rmk
    (LET ((LINES (GIT-COMMAND (CONCAT "git log --date=rfc -- " (GIT-REPO-FILENAME FILE PROJECT))
                      T NIL (GIT-GET-PROJECT PROJECT)))
          VAL)
         [FOR L COMMIT COMMENTS POS IN (REVERSE LINES) UNLESS (ZEROP (NCHARS L))
            DO (IF (STRPOS "commit " L 1 NIL 1)
                   THEN (CL:WHEN COMMENTS
                             (SETQ COMMIT (NCONC1 COMMIT (CONS 'Comments COMMENTS)))
                             (SETQ COMMENTS NIL))
                        (PUSH VAL (CONS (LIST 'commit (SUBSTRING L 8))
                                        COMMIT))
                        (SETQ COMMIT NIL)
                   ELSEIF (EQ (CHARCODE SPACE)
                              (CHCON1 L))
                     THEN (PUSH COMMENTS (OR (SUBSTRING L (FIND I FROM 2 SUCHTHAT (NEQ (CHARCODE SPACE)
                                                                                       (NTHCHARCODE L I)))
                                                        -1)
                                             T))
                   ELSE (PUSH COMMIT (LIST [SUBATOM L 1 (OR (SETQ POS (SUB1 (STRPOS ": " L 1]
                                                            (SUBSTRING L (FIND I FROM (IPLUS 2 POS)
                                                                               SUCHTHAT (NEQ (CHARCODE SPACE)
                                                                                             (NTHCHARCODE L I)))
                                                            -1]
         (CL:WHEN PRINT (GIT-PRINT-FILE-HISTORY VAL))
         (CONS (GIT-REPO-FILENAME FILE PROJECT)
               VAL])
```

(GIT-PRINT-FILE-HISTORY

```
  [LAMBDA (COMMITS AUTHORS)                                       ; Edited  2-Jul-2022 00:21 by rmk
    (PRINTOUT T (CAR COMMITS)
           T)
    (FOR C AU IN (CDR COMMITS) EACHTIME (SETQ AU (CADR (ASSOC 'Author C)))
       WHEN (OR (NULL AUTHORS)
                (THEREIS A INSIDE AUTHORS SUCHTHAT (STRPOS A AU 1 NIL NIL NIL UPPERCASEARRAY)))
       DO (PRINTOUT T 5 (CAAR C)
                ": "
                (CADAR C)
                T)
          (FOR X IN (CDR C) DO (PRINTOUT T 10 (CAR X)
                                       ": ")
                               (IF (EQ (CAR X)
                                       'Comments)
                                   THEN (FOR CC (POS _ (POSITION T)) IN (CDR X)
                                             DO (IF (EQ CC T)
                                                    THEN (TERPRI T)
                                                  ELSE (PRINTOUT T .TAB0 POS CC)))
                                 ELSE (PRINTOUT T (CADR X)))
                               (TERPRI T))
          (TERPRI T])
```

(GIT-FETCH

```
  [LAMBDA (PROJECT)                                               ; Edited  8-Jul-2022 10:32 by rmk
    (GIT-COMMAND "git fetch" T NIL PROJECT])

)

;; Differences

(DEFINEQ
```

(GIT-BRANCH-DIFF

```
  [LAMBDA (BRANCH1 BRANCH2 PROJECT)

    ;; Edited 10-Jun-2024 16:43 by mth

    ;; Edited  2-May-2024 11:28 by mth

    ;; Edited 29-Sep-2022 10:52 by rmk

    ;; Edited 12-Sep-2022 14:13 by rmk

    ;; Edited 17-Jul-2022 09:36 by rmk

    ;; Edited  4-Jun-2022 20:43 by rmk

    ;; Edited  9-May-2022 16:21 by rmk: returns an ALIST that classifies how the files in BRANCH1 and BRANCH2 differ (changed, renamed, added,
    ;; deleted, copied).

    ;; Edited  6-May-2022 14:04 by rmk: returns an ALIST that classifies how the files in BRANCH1 and BRANCH2 differ (changed, renamed, added,
    ;; deleted, copied).

    ;; This returns an ALIST that classifies how the files in BRANCH1 and BRANCH2 differ (changed, renamed, added, deleted, copied).

    (CL:UNLESS BRANCH1
```

```
        (SETQ BRANCH1 (GIT-MAINBRANCH PROJECT)))
    (CL:UNLESS BRANCH2
        (SETQ BRANCH2 (GIT-MAINBRANCH PROJECT)))
    (GIT-REMOTE-UPDATE NIL PROJECT)
```

;; We don't use GIT-COMMAND because we want to deal with the warning messages here, to give the option of increasing the rename limit..

```
    (PROG (POS LIMIT ERRORFILE RLINES ELINES RESULTFILE)
      RETRY
```

;; Nick previously suggested: "git diff --name-status -C --find-copies-harder <merge> branch1", but that brought in too many files. The
;; merge-base seems to match the Git desktop.

```
          (SETQ RESULTFILE (GIT-COMMAND-TO-FILE (CONCAT "git diff -C --find-copies-harder $(git merge-base %""
                                                        BRANCH1 "%" %"" BRANCH2 "%") %"" BRANCH2 "%"
                                                        --name-status")
                                    PROJECT))
          (SETQ ELINES NIL)
          (SETQ RLINES NIL)
          (CL:WHEN (LISTP RESULTFILE)
              (SETQ ERRORFILE (CADR RESULTFILE))
              (SETQ ELINES (GIT-RESULT-TO-LINES ERRORFILE T))
              (DELFILE ERRORFILE)
              (SETQ RESULTFILE (CAR RESULTFILE)))
          (SETQ RLINES (GIT-RESULT-TO-LINES RESULTFILE T))
          (DELFILE RESULTFILE)
          (CL:WHEN ELINES
              (if [AND (STRPOS "warning: inexact rename detection was skipped due to too many files."
                               (CAR ELINES)
                               1)
                       (SETQ LIMIT (FIXP (SUBATOM (CADR ELINES)
                                                  (STRPOS " at least " (CADR ELINES)
                                                         1 NIL NIL T)
                                                  (SUB1 (STRPOS " and retry " (CADR ELINES]
                  then (PRINTOUT T 3 "** For accurate branch differences, " "diff.renameLimit must be
                                     increased")
                       (SELECTQ (AND LIMIT (ASKUSER NIL 'N (CONCAT "      Should I increase the global limit to
" (add LIMIT 1)
                                                                  " and try again? ")))
                                (Y (GIT-COMMAND (CONCAT "git config --global diff.renameLimit " LIMIT)
                                                NIL NIL PROJECT)
                                   (GO RETRY))
                                (ERROR "Incomplete branch differences" (LIST BRANCH1 BRANCH2)))
                  else (for L in ELINES do (PRINTOUT T L T))))
          (RETURN (SORT (for (L FN) in RLINES
                           collect (SELCHARQ (CHCON1 L)
                                             (A (CL:IF (EQ (CHARCODE TAB)
                                                           (NTHCHARCODE L 2))
                                                       (LIST 'ADDED (SETQ FN (SUBSTRING L 3)))
                                                       (ERROR "ADDED NOT RECOGNIZED" L)))
                                             (D (CL:IF (EQ (CHARCODE TAB)
                                                           (NTHCHARCODE L 2))
                                                       (LIST 'DELETED (SETQ FN (SUBSTRING L 3)))
                                                       (ERROR "DELETED NOT RECOGNIZED" L)))
                                             (M (CL:IF (SETQ POS (STRPOS "     " L))
                                                       [LIST 'CHANGED (SETQ FN (SUBSTRING L (ADD1 POS]
                                                       (ERROR "CHANGED NOT RECOGNIZED" L)))
                                             (C (if (AND (EQ (CHARCODE TAB)
                                                             (NTHCHARCODE L 5))
                                                         (SETQ POS (STRPOS "    " L 7)))
                                                    then (LIST 'COPIED (SETQ FN (SUBSTRING L 6 (SUB1 POS)))
                                                               (OR (FIXP (SUBATOM L 2 4))
                                                                   (HELP "C without a number" L)))
                                                  else (HELP "COPY NOT RECOGNIZED" L)))
                                             (R (if (AND (EQ (CHARCODE TAB)
                                                             (NTHCHARCODE L 5))
                                                         (SETQ POS (STRPOS "    " L 7)))
                                                    then (LIST 'RENAMED (SETQ FN (SUBSTRING L 6 (SUB1 POS)))
                                                               (SUBSTRING L (ADD1 POS))
                                                               (OR (FIXP (SUBATOM L 2 4))
                                                                   (HELP "R without a number" L)))
                                                  else (HELP "RENAME NOT RECOGNIZED" L)))
                                             (w (CL:UNLESS (STRPOS "warning: " L 1)
                                                           (HELP "UNRECOGNZED GIT LINE" L))
                                                (CL:UNLESS (EQ 'Y (ASKUSER NIL NIL (CONCAT L " Ignore remaining files?
")))
                                                           (ERROR!)))
                                             (HELP "Unrecognized git-diff code " L))
                           unless (STREQUAL ".git/" (SUBSTRING FN 1 5)))
                       T])
```

(GIT-COMMIT-DIFFS

```
  [LAMBDA (BRANCH1 BUTNOTBRANCH2 PROJECT)                    ; Edited  2-May-2024 11:24 by mth
                                                             ; Edited 26-Jun-2022 13:32 by rmk
                                                             ; Edited  7-May-2022 23:48 by rmk
                                                             ; Edited  2-May-2022 13:45 by rmk
```

;; Returns the identifiers for commits in BRANCH1 but not in BUTNOTBRANCH2

```
    (GIT-COMMAND (CONCAT "git log --format=%"%%h%" %"" BRANCH1 "%" %"^" BUTNOTBRANCH2 "%"")
           NIL NIL PROJECT])
```

(GIT-BRANCH-RELATIONS

```
  [LAMBDA (BRANCHES BRANCH2 STRIPWHERE PROJECT)                    ; Edited  4-Aug-2022 10:38 by rmk
                                                                   ; Edited 29-May-2022 21:59 by rmk
                                                                   ; Edited  9-May-2022 16:12 by rmk
    (SETQ PROJECT (GIT-GET-PROJECT PROJECT))
    ;; Returns a pair (SUPERSETS EQUALS), where each item in SUPERSETS is a list of the form (B0 B1 B2...) where each Bi is a superset of Bj for i
    ;; < j and EQUALS is a list of branch equivalence classes.
    (LET ((MAIN (GIT-MAINBRANCH PROJECT)))
         (CL:WHEN STRIPWHERE
             (SETQ MAIN (STRIPWHERE MAIN)))
         (for DTAIL D1 MORE1 MORE2 SUPERSETS EQUALS on (for B in BRANCHES
                                                          collect (CL:WHEN STRIPWHERE
                                                                       (SETQ B (STRIPWHERE B)))
                                                                  (CONS B (GIT-COMMIT-DIFFS B MAIN PROJECT)))
           do ;; For each branch we now have the list of commit identifiers (hexstrings) that they do not share with the main branch.
              (SETQ D1 (CAR DTAIL))
              [for D2 in (CDR DTAIL)
                 do (CL:WHEN (EQUAL (CDR D1)
                                    (CDR D2))                      ; Unlikely
                             (push [CDR (OR (ASSOC (CAR D1)
                                                   EQUALS)
                                            (CAR (push EQUALS (CONS (CAR D1]
                                   (CAR D2))
                             (GO $$ITERATE))
                    (SETQ MORE2 (MEMBER (CADR D1)
                                        (CDR D2)))                 ; The most recent commit of D1 is in D2
                    (SETQ MORE1 (MEMBER (CADR D2)
                                        (CDR D1)))
                    (if MORE2
                        then (CL:UNLESS MORE1
                                  (push [CDR (OR (ASSOC (CAR D2)
                                                        SUPERSETS)
                                                 (CAR (push SUPERSETS (CONS (CAR D2]
                                        (CAR D1)))
                      elseif MORE1
                        then (push [CDR (OR (ASSOC (CAR D1)
                                                   SUPERSETS)
                                            (CAR (push SUPERSETS (CONS (CAR D1]
                                   (CAR D2]
           finally ;; Sort the supersets so that the larger ones come before the smaller ones
                   (CL:WHEN STRIPWHERE
                       [SETQ SUPERSETS (for S in SUPERSETS collect (for SS in S collect (STRIPWHERE SS]
                       [SETQ EQUALS (for S in EQUALS collect (for SS in S collect (STRIPWHERE SS])
                   [for S in SUPERSETS do (change (CDR S)
                                                  (SORT DATUM (FUNCTION (LAMBDA (B1 B2)
                                                                          (OR (MEMB B2 (CDR (ASSOC B1 SUPERSETS)))
                                                                              (NOT (MEMB B1 (CDR (ASSOC B2
                                                                                                        SUPERSETS]
                   [for E in EQUALS do (change (CDR E)
                                               (if (MEMB MAIN (CDR E))
                                                   then (CONS MAIN (DREMOVE MAIN (SORT DATUM)))
                                                 else (SORT DATUM]
                   (RETURN (LIST SUPERSETS EQUALS])

)

;;
;; Branches

(DEFINEQ

(GIT-BRANCH-NUM
  [LAMBDA (BRANCH INITS)                                           ; Edited 19-May-2022 19:11 by rmk
    ;; Returns nnn if BRANCH is ({local|origin}/)INITSnnn(-xxxx)
    (CL:UNLESS INITS
        (SETQ INITS (GIT-INITIALS)))
    (LET (NPOS (SPOS (OR (STRPOS "/" BRANCH 1 NIL NIL T)
                         1)))
         (CL:WHEN (SETQ NPOS (STRPOS INITS BRANCH SPOS NIL NIL T UPPERCASEARRAY))
             [NUMBERP (SUBATOM BRANCH NPOS (SUB1 (OR (STRPOS "-" BRANCH NPOS)
                                                     0]))])

(GIT-CHECKOUT
  [LAMBDA (BRANCH PROJECT)                                         ; Edited 12-Jun-2024 22:44 by mth
                                                                   ; Edited  2-May-2024 11:17 by mth
                                                                   ; Edited  7-Jul-2022 20:21 by rmk
```

```
                                                                    ; Edited  9-May-2022 15:12 by rmk
                                                                    ; Edited  7-May-2022 23:51 by rmk
                                                                    ; Edited  2-Nov-2021 22:40 by rmk:
    (CL:UNLESS BRANCH
        (SETQ BRANCH (GIT-MAINBRANCH PROJECT)))
    (LET ((CURRENTBRANCH (GIT-WHICH-BRANCH PROJECT T)))
         [SETQ CURRENTBRANCH (SUBSTRING CURRENTBRANCH (ADD1 (STRPOS "/" CURRENTBRANCH]
         (CL:UNLESS [STRING.EQUAL CURRENTBRANCH (SUBSTRING BRANCH (ADD1 (OR (STRPOS "/" BRANCH)
                                                                            0]
             (GIT-COMMAND (CONCAT "git checkout %"" BRANCH "%"")
                    NIL T PROJECT)
             (CAR (GIT-COMMAND (CONCAT "git pull")
                          NIL T PROJECT)))
         BRANCH])
```

(GIT-WHICH-BRANCH

```
  [LAMBDA (PROJECT ALL)                                             ; Edited 12-Jun-2024 12:57 by mth
                                                                    ; Edited  7-May-2022 22:41 by rmk
    ;; Returns the current (local) branch in PROJECT
    (MKATOM (CONCAT "local/" (CAR (GIT-COMMAND "git rev-parse --abbrev-ref HEAD" ALL NIL PROJECT])
```

(GIT-MAKE-BRANCH

```
  [LAMBDA (NAME TITLESTRING PROJECT)                                ; Edited 12-Jun-2024 22:47 by mth
                                                                    ; Edited  2-May-2024 11:24 by mth
                                                                    ; Edited 18-Jul-2022 21:45 by rmk
                                                                    ; Edited 19-May-2022 17:57 by rmk
                                                                    ; Edited  9-May-2022 15:13 by rmk
    ;;  The new branch is directly under the currently checked out branch.  Maybe it should always make it under the main branch?
    ;; This makes a new branch with name NAME: TITLESTRING, or just NAME if TITLESTRING is not given.
    ;; (GIT-MAKE-BRANCH) makes and checks out the next initialsn branch.
    (CL:UNLESS NAME
        (SETQ NAME (GIT-MY-NEXT-BRANCH PROJECT)))
    (CL:WHEN TITLESTRING
        ;; Git branch names can't contain spaces or colons
        ;; mth: Notice that this is only dealing with spaces. There are other "troublesome" characters beyond colon, as well.
        [SETQ TITLESTRING (CONCATCODES (for I C from 1 while (SETQ C (NTHCHARCODE TITLESTRING I))
                                          collect (if (EQ C (CHARCODE SPACE))
                                                      then (CHARCODE -)
                                                    else C]
        (SETQ NAME (CONCAT NAME "--" TITLESTRING)))
    (LET ((UNDER (GIT-WHICH-BRANCH PROJECT T))
          RESULT)
         (if (EQ 'Y (ASKUSER NIL 'N (CONCAT "Branch " NAME " will be created under " UNDER ". Is that OK? ")))
             then (SETQ RESULT (GIT-COMMAND (CONCAT "git checkout -b %"" NAME "%"")
                                       NIL NIL PROJECT))
                  (if (STREQUAL (CAR RESULT)
                                (CONCAT "Switched to a new branch '" NAME "'"))
                      then (CONCAT (CAR RESULT)
                                   " under " UNDER)
                    elseif (STREQUAL (CAR RESULT)
                                     (CONCAT "fatal: A branch named '" NAME "' already exists."))
                      then (ERROR NAME "already exists")
                      else (HELP "Unexpected git result" RESULT))
           else (PRINTOUT T "New branch not created" T)
                NIL])
```

(GIT-BRANCHES

```
  [LAMBDA (WHERE PROJECT EXCLUDEMERGED)                             ; Edited 12-Jun-2024 12:46 by mth
                                                                    ; Edited  2-May-2024 11:26 by mth
                                                                    ; Edited  9-Aug-2022 10:45 by rmk
                                                                    ; Edited 18-Jul-2022 08:11 by rmk
                                                                    ; Edited  8-Jul-2022 10:33 by rmk
                                                                    ; Edited 23-May-2022 14:25 by rmk
                                                                    ; Edited 19-May-2022 10:06 by rmk
                                                                    ; Edited  9-May-2022 14:10 by rmk
                                                                    ; Edited  7-May-2022 23:29 by rmk
    (SETQ PROJECT (GIT-GET-PROJECT PROJECT))
    ;; Strips of the "* " that indicates the current branch and the 2-space padding on other branches.  Packs local/ on to local branches
    (LET ([LOCAL (CL:WHEN (MEMB (U-CASE WHERE)
                                '(NIL ALL LOCAL))
                          [for B in (GIT-COMMAND "git branch" T NIL PROJECT)
                             collect (SUBATOM B 3 (SUB1 (OR (STRPOS " -> " B)
                                                            0]
          [REMOTE (CL:WHEN (MEMB (U-CASE WHERE)
                                 '(NIL ALL REMOTE T))
                           [for B in (GIT-COMMAND "git branch -r" T NIL PROJECT)
                              collect (SUBATOM B 3 (SUB1 (OR (STRPOS " -> " B)
                                                             0]
          BRANCHES)
```

```
        (SETQ BRANCHES (UNION LOCAL REMOTE))
        (CL:WHEN (thereis B in BRANCHES suchthat (STRPOS "HEAD detached" B))
               (PRINTOUT T "Execute %"git gc%" to eliminate a branch with a detached HEAD" T))
        (CL:WHEN EXCLUDEMERGED
            (SETQ BRANCHES (for B (MAINBRANCH _ (GIT-MAINBRANCH PROJECT 'LOCAL)) in BRANCHES
                              when (EQUAL (GIT-COMMAND (CONCAT "git merge-base %"" B "%" %"" MAINBRANCH "%""))
                                          )
                                          (GIT-COMMAND (CONCAT "git rev-parse %"" B "%""))))
                              collect B)))
        (SORT BRANCHES])
```

(GIT-BRANCH-EXISTS?

```
  [LAMBDA (BRANCH NOERROR PROJECT EXCLUDEMERGED)                    ; Edited  2-May-2024 13:00 by mth
                                                                    ; Edited 19-May-2022 10:10 by rmk
```

;; Returns the canonical name of the branch (xxx or origin/xxx) depending on whether BRANCH is local/xxx or origin/xxx

```
    (LET [(WHERE (if (STRPOS "origin/" BRANCH 1 NIL T)
                     then 'REMOTE
                   elseif (STRPOS "local/" BRANCH 1 NIL T)
                     then [SETQ BRANCH (SUBSTRING BRANCH (ADD1 (STRPOS "/" BRANCH 1]
                          'LOCAL]
          (if (CAR (MEMB (MKATOM BRANCH)
                         (GIT-BRANCHES WHERE PROJECT EXCLUDEMERGED)))
            elseif (NOT NOERROR)
              then (ERROR "Unknown branch" BRANCH])
```

(GIT-PICK-BRANCH

```
  [LAMBDA (BRANCHES TITLE)                                          ; Edited  6-Jul-2023 22:31 by rmk
                                                                    ; Edited 30-Jun-2023 16:58 by rmk
                                                                    ; Edited 18-May-2022 13:44 by rmk
    (MENU (GIT-BRANCH-MENU BRANCHES (OR TITLE (CONCAT (LENGTH BRANCHES)
                                                      " branches"])
```

(GIT-BRANCH-MENU

```
  [LAMBDA (BRANCHES TITLE PIN?)                                     ; Edited  1-May-2024 14:36 by rmk
                                                                    ; Edited  6-Jul-2023 22:31 by rmk
                                                                    ; Edited 30-Jun-2023 16:58 by rmk
                                                                    ; Edited 18-May-2022 13:44 by rmk
    (CL:WHEN (SETQ BRANCHES (MKLIST BRANCHES))
        (CL:WHEN PIN?
           [SETQ BRANCHES (APPEND BRANCHES '(("      Pin menu" 'PinMenu])
        (create MENU
               TITLE _ (OR TITLE (CONCAT (LENGTH BRANCHES)
                                         " branches"))
               ITEMS _ BRANCHES
               MENUFONT _ DEFAULTFONT
               WHENSELECTEDFN _ (FUNCTION GIT-BRANCH-WHENSELECTEDFN)))])
```

(GIT-BRANCH-WHENSELECTEDFN

```
  [LAMBDA (ITEM)                                                    ; Edited 11-May-2024 11:05 by rmk
                                                                    ; Edited  1-May-2024 18:17 by rmk
                                                                    ; CAR is git key, 4th is project
```

;; This executes the comparison in the current TTY window, either by stuffing the command there or by evaluating there. There probably should be ;; a check to make sure that the TTY is in fact an executive--if not, maybe this should be a no-op. Better than getting the comparison form in the ;; middle of anther SEDIT or TEDIT.

;; This could also execute in the mouse process, where the menu is clicked. But in that case a terminal window pops up with the header lines of ;; the compare, and that seems a nuisance.

```
    (LET [(PR (CAR (LAST ITEM]
         (if [AND NIL (PROGN (GETMOUSESTATE)
                             (EQ 'MIDDLE (DECODEBUTTONS]
             then (LET [(POS (ADD1 (STRPOS "#" (CAR ITEM]
                       (ShellBrowse (fetch PRURL of PR)))
           elseif (PROGN T)
             then ;; PROGN because DWIM is screwed up

                  ;; The COPYINSERT causes the compare to run in the TTY process, by stuffing the characters in the input line. Somehow it
                  ;; executes even if the parens are not there, but that looks funny. But it also works if I stuff the parens on both sides.

                  (BKSYSBUF '%()
                  [COPYINSERT `(GIT-PR-COMPARE ,(CADR ITEM)
                                               ',(fetch PRPROJECT of PR]
                  (BKSYSBUF '%))
             else ;; This puts the print out after the next event number in the TTY window, unfortunately. We go to the default font so we don't get
                  ;; TTYIN's input bold for this.

                  (PROCESS.EVAL (TTY.PROCESS)
                         `(RESETLST
                              [RESETSAVE (DSPFONT DEFAULTFONT T)
                                     '(PROGN (DSPFONT OLDVALUE T])])
```

(GIT-PULL-REQUESTS

```
  [LAMBDA (INCLUDEDRAFTS PROJECT)                                    ; Edited 20-May-2024 22:12 by rmk
                                                                     ; Edited 13-May-2024 18:59 by rmk
                                                                     ; Edited 11-May-2024 10:51 by rmk
                                                                     ; Edited  1-May-2024 09:23 by rmk
                                                                     ; Edited  8-Aug-2022 13:12 by rmk
                                                                     ; Edited  4-Aug-2022 19:01 by rmk
                                                                     ; Edited 17-Jul-2022 11:12 by rmk
                                                                     ; Edited  9-May-2022 16:54 by rmk
    ;; Returns a list of PULLREQUEST records, one for each pull request ; Edited 25-Feb-2022 09:26 by rmk
    (CL:UNLESS (EQ 0 (PROCESS-COMMAND "command -v gh"))
           (ERROR "gh must be installed in order to enumerate pull requests:"))
    (LET [(JPARSE (JSON-PARSE (CAR (GIT-COMMAND "gh pr list --json
                                                number,headRefName,title,isDraft,reviewDecision,url,headRepository,hea
dRepositoryOwner" T NIL PROJECT]
          (for JSOBJ DRAFT PR in (SELECTQ (CAR JPARSE)
                                    (ARRAY (CDR JPARSE))
                                    (OBJECT JPARSE)
                                    (ERROR "UNRECOGNIZED PRC LIST FROM GIT" JPARSE))
             eachtime [SETQ DRAFT (EQ 'true (JSON-GET JSOBJ 'isDraft] when (OR INCLUDEDRAFTS (NOT DRAFT))
             collect [SETQ PR (create PULLREQUEST
                                    PRNUMBER _ (JSON-GET JSOBJ 'number)
                                    PRNAME _ (JSON-GET JSOBJ 'headRefName)
                                    PRDESCRIPTION _ (JSON-GET JSOBJ 'title)
                                    PRSTATUS _ (CL:IF DRAFT
                                                   'D
                                                   (CL:IF (STREQUAL "REVIEW_REQUIRED" (JSON-GET JSOBJ
                                                                                              'reviewDecision))
                                                       " "
                                                       'A))
                                    PRPROJECT _ PROJECT
                                    PRURL _ (JSON-GET JSOBJ 'url)
                                    PRLOGIN _ (JSON-GET JSOBJ '(headRepositoryOwner login]
                  (CL:WHEN (STRPOS ":" (fetch (PULLREQUEST PRNAME) of PR))
                      ;; From Nick:  Git commands to bring install and deal with the remotes:
                      ;; git remote add [PRLOGIN] https://github.com/[PRLOGIN]/[PROJECT]
                      ;;        (project in lower-case)
                      ;; git remote update [PRLOGIN]
                      (PRINTOUT T "Ignoring PR for forked repo %%%" #" (JSON-GET JSOBJ 'number)
                             " "
                             (fetch (PULLREQUEST PRNAME) of PR)
                             "%"" T)
                      (GO $$ITERATE))
                  PR])
```

(GIT-SHORT-BRANCH-NAME

```
  [LAMBDA (BRANCH)                                                   ; Edited 22-May-2022 22:36 by rmk
    ;; Reduces rmk29--xxxxx to rmk29 for display
    (SUBSTRING BRANCH 1 (SUB1 (OR (STRPOS "--" BRANCH 1)
                                  0])
```

(GIT-LONG-NAME

```
  [LAMBDA (BRANCH WHERE PROJECT EXCLUDEMERGED)                       ; Edited 24-May-2022 17:49 by rmk
    ;; Allows short-hand reference to branch:  rmk40 will return rmk40--xyz
    (FIND B IN (GIT-BRANCHES WHERE PROJECT EXCLUDEMERGED) SUCHTHAT (STRPOS BRANCH B])
```

(GIT-PRC-BRANCHES

```
  [LAMBDA (DRAFT PROJECT PRS)                                        ; Edited 13-May-2024 19:30 by rmk
                                                                     ; Edited 11-May-2024 10:52 by rmk
                                                                     ; Edited  1-May-2024 21:06 by rmk
                                                                     ; Edited  1-Apr-2024 17:09 by rmk
                                                                     ; Edited  8-Aug-2022 18:15 by rmk
                                                                     ; Edited  4-Aug-2022 18:55 by rmk
                                                                     ; Edited  9-Jul-2022 19:01 by rmk
                                                                     ; Edited 16-May-2022 19:44 by rmk
    ;; This converts each PR into a list of a form that can be used as a menu item.  PROJECT is added at the end, beyond what is interpreted by the
    ;; menu machinery.  Maybe the 4th item should be the entire PR, with PROJECT inside it.
    (CL:UNLESS PRS
        (SETQ PRS (GIT-PULL-REQUESTS T PROJECT)))
    (CL:WHEN PRS
        (LET ((RELATIONS (GIT-BRANCH-RELATIONS (for PR in PRS collect (GITORIGIN (fetch PRNAME of PR)))
                                 NIL T PROJECT)))
             (SORT (for PR REL LABEL PRNAME STATUS (SUPERSETS _ (CAR RELATIONS))
                        (EQUALS _ (CADR RELATIONS)) in PRS
                   eachtime (SETQ PRNAME (fetch PRNAME of PR))
                         (SETQ LABEL (CONCAT "#" (fetch (PULLREQUEST PRNUMBER) of PR)
                                             " "
                                             (if [SETQ REL (CAR (CDR (SASSOC PRNAME SUPERSETS]
                                                 then (CONCAT PRNAME " > " REL)
```

```
                                                    elseif [SETQ REL (CAR (CDR (SASSOC PRNAME EQUALS]
                                                      then (CONCAT PRNAME " = " REL)
                                                    else PRNAME)))
                                (SETQ STATUS (fetch PRSTATUS of PR))
                            when (SELECTQ DRAFT
                                     (DRAFTS (EQ STATUS 'D))
                                     (NODRAFTS (NEQ STATUS 'D))
                                     T)
                            collect (LIST (CONCAT " " STATUS " " LABEL)
                                          (GITORIGIN PRNAME)
                                          (CONCAT " " STATUS " #" (fetch PRNUMBER of PR)
                                                  " "
                                                  (fetch PRDESCRIPTION of PR))
                                          NIL PR))
                        T))))])

)

;; My branches

(DEFINEQ

(GIT-MY-CURRENT-BRANCH
  [LAMBDA (PROJECT INITS)                                   ; Edited 19-May-2022 19:13 by rmk
    (CL:UNLESS INITS
        (SETQ INITS (GIT-INITIALS)))
    (FOR B IN (GIT-MY-BRANCHES PROJECT NIL INITS) LARGEST (OR (GIT-BRANCH-NUM B INITS)
                                                              0])

(GIT-MY-BRANCHP
  [LAMBDA (BRANCH PROJECT)                                  ; Edited 12-Jun-2024 22:48 by mth
                                                            ; Edited 19-May-2022 17:44 by rmk
                                                            ; Edited 19-Jan-2022 13:22 by rmk
    ;; Returns n if BRANCH is INITIALSn (local or origin), possibly followed by a trailing comment after hyphen.
    (CL:UNLESS BRANCH
        (SETQ BRANCH (GIT-WHICH-BRANCH PROJECT T)))
    (GIT-BRANCH-NUM (OR BRANCH (GIT-WHICH-BRANCH PROJECT T])

(GIT-MY-NEXT-BRANCH
  [LAMBDA (PROJECT)                                         ; Edited 19-May-2022 14:08 by rmk
                                                            ; Edited  8-Jan-2022 09:43 by rmk
    ;; Figures out the number of my next incremental branch would be.
    (PACK* (GIT-INITIALS)
           (ADD1 (OR (GIT-MY-BRANCHP (GIT-MY-CURRENT-BRANCH PROJECT)
                                     PROJECT)
                     0])

(GIT-MY-BRANCHES
  [LAMBDA (PROJECT EXCLUDEMERGED INITS)                     ; Edited 19-May-2022 19:10 by rmk
    ;; This returns only local branch names:  xyzn and not origin/xyzn or local/xyzn
    ;; If INITIALS is xyz or xyz:, returns xyzn where xyzn is a branch and n is greater than m for all other branches xyzm.  xyzn may not be be the
    ;; current branch.
    ;; The return list is sorted so that lower n's come before later n's.  The last element is my current branch
    (CL:UNLESS INITS
        (SETQ INITS (GIT-INITIALS)))
    (FOR B IPOS IN (GIT-BRANCHES 'LOCAL PROJECT EXCLUDEMERGED)
       WHEN [AND (SETQ IPOS (STRPOS INITS B 1 NIL NIL NIL UPPERCASEARRAY))
                 (OR (EQ IPOS 1)
                     (EQ (CHARCODE /)
                         (NTHCHARCODE B (SUB1 IPOS]
       COLLECT (CONS B (GIT-BRANCH-NUM B INITS))
       FINALLY
            ;; We expect a branch beginning with INITS  rmk is of the form "rmknnn" or "rmknnn--somestring".  If so, we want to sort b the number.
            ;; If not, sort alphabetically at the end, with numbered ones first.
            (RETURN (FOR B IN [SORT $$VAL (FUNCTION (LAMBDA (X Y)
                                                    (IF (CDR X)
                                                        THEN (IF (CDR Y)
                                                                 THEN (ILESSP (CDR X)
                                                                              (CDR Y))
                                                               ELSE T)
                                                      ELSEIF (NOT (CDR Y))
                                                        THEN (ALPHORDER (CAR X)
                                                                        (CAR Y]
                             COLLECT (CAR B])

)

;;
```

;; Worktrees

```
(DEFINEQ
```

(GIT-ADD-WORKTREE

```
  [LAMBDA (BRANCH REMOTEONLY PROJECT)                        ; Edited  2-May-2024 11:25 by mth
                                                             ; Edited  9-May-2022 14:17 by rmk

    (SETQ BRANCH (GITORIGIN BRANCH (NOT REMOTEONLY)))
    (CL:UNLESS (OR (GIT-BRANCH-EXISTS? BRANCH T PROJECT)
                   (GIT-BRANCH-EXISTS? BRANCH T PROJECT))
           (ERROR BRANCH "is not a git branch"))
    (CL:WHEN (STRING-EQUAL BRANCH (GIT-WHICH-BRANCH PROJECT))
           (ERROR BRANCH "is the current branch"))
    (LET (LINES LOCALBRANCH)
         (SETQ LINES (GIT-COMMAND (if (EQ 1 (STRPOS "origin/" BRANCH))
                                      then [SETQ LOCALBRANCH (SUBSTRING BRANCH (CONSTANT (ADD1 (NCHARS
                                                                                                "origin/"
                                                                                                ]
                                                  (CONCAT "git worktree add --guess-remote " (WORKTREEDIR LOCALBRANCH
                                                                                                  PROJECT)
                                                          " %"" BRANCH "%"")
                                                else (CONCAT "git worktree add " (WORKTREEDIR BRANCH)
                                                          " %"" BRANCH "%""))
                                  NIL NIL PROJECT))
         (CL:UNLESS (STRPOS "Preparing worktree" (CAR LINES)
                            1 NIL T)
                (ERROR "Could not create worktree for " BRANCH))
         BRANCH])
```

(GIT-REMOVE-WORKTREE

```
  [LAMBDA (BRANCH PROJECT)                                   ; Edited  9-May-2022 16:22 by rmk
                                                             ; Edited 17-Nov-2021 10:02 by rmk:

    (GIT-BRANCH-EXISTS? BRANCH NIL PROJECT)
    (LET ((DIR (WORKTREEDIR BRANCH PROJECT))
          LINES)
         (SETQ LINES (GIT-COMMAND (CONCAT "git worktree remove " DIR)
                                  NIL NIL PROJECT))
         (CL:WHEN (STRPOS "fatal: " (CAR LINES)
                          1 NIL T)
                (ERROR "Could not remove worktree for " BRANCH)) (* (DELFILE (CONCAT PATH "/.DS_Store"))
                                                                    (GIT-COMMAND (CONCAT "rmdir " DIR) NIL NIL PROJECT))
         BRANCH])
```

(GIT-LIST-WORKTREES

```
  [LAMBDA NIL                                                ; Edited 12-Dec-2021 12:13 by rmk
                                                             ; Edited 19-Nov-2021 18:53 by rmk:
```

;; The git command tells us what the clone thinks about it, but then we look to see what is actually in our worktrees directory, to make sure that the ;; subdirectory wasn't deleted in a wy that the clone didn't know about.

```
    (SORT (FOR L POS IN (GIT-COMMAND "git worktree list")
             WHEN (AND (SETQ POS (STRPOS "/worktrees/" L NIL NIL NIL T))
                       (STRPOS "(detached HEAD)" L))
             COLLECT (SETQ L (SUBSTRING L POS))
                   (SUBATOM L 1 (SUB1 (STRPOS " " L])
```

(WORKTREEDIR

```
  [LAMBDA (BRANCH PROJECT)                                   ; Edited  9-May-2022 00:04 by rmk
                                                             ; Edited 18-Jan-2022 15:02 by rmk
                                                             ; Edited 25-Nov-2021 08:49 by rmk:
                                                             ; Edited 19-Nov-2021 20:56 by rmk:
                                                             ; Edited 17-Nov-2021 10:00 by rmk:

    (CONCAT (FETCH GITHOST OF PROJECT)
           "../worktrees/"
           (IF BRANCH
               THEN "/"
             ELSE ""])

)
```

;;

;; Comparisons

```
(DEFINEQ
```

(GIT-GET-DIFFERENT-FILES

```
  [LAMBDA (BRANCH1 BRANCH2 DIR1 DIR2 PROJECT)
    (DECLARE (USEDFREE FROMGITN))
```

;; Edited 12-Sep-2022 14:58 by rmk

;; Edited 21-May-2022 23:38 by rmk

;; Edited 9-May-2022 14:17 by rmk: Ask git for the files that differ between the branches, copy those files down to local DIR1 and DIR2, return the ;; directories and a list of (dir1-file1 file2) mappings for renamed and copied files.

```
;; Edited  6-May-2022 08:26 by rmk: Ask git for the files that differ between the branches, copy those files down to local DIR1 and DIR2, return the
;; directories and a list of (dir1-file1 file2) mappings for renamed and copied files.

(SETQ PROJECT (GIT-GET-PROJECT PROJECT))
(SETQ BRANCH1 (GIT-BRANCH-EXISTS? BRANCH1 NIL PROJECT))
(SETQ BRANCH2 (GIT-BRANCH-EXISTS? BRANCH2 NIL PROJECT))
(LET (MAPPINGS FROMGIT (DIFFS (GIT-BRANCH-DIFF BRANCH1 BRANCH2 PROJECT)))
     (CL:WHEN DIFFS
         (SETQ FROMGIT (PACK* '{FROMGIT (add FROMGITN 1)
                              '}))
         (PSEUDOHOST FROMGIT (CONCAT "{CORE}<" (fetch PROJECTNAME of PROJECT)
                                     ">"
                                     (DATE)
                                     ">"))

         ;; UNSLASHIT because CORE doesn't know about slash

         (CL:UNLESS DIR1
             (SETQ DIR1 (CONCAT FROMGIT "<" (UNSLASHIT BRANCH1)
                                ">")))
         (CL:UNLESS DIR2
             (SETQ DIR2 (CONCAT FROMGIT "<" (UNSLASHIT BRANCH2)
                                ">")))
         (for D in DIFFS
            do (SELECTQ (CAR D)
                    (ADDED                                  ; Shouldn't exist in BRANCH2, should exist in BRANCH1, but
                                                            ; maybe ADDED and DELETED are mixed up?
                           (SETQ D (CADR D))
                           (OR (GIT-GET-FILE BRANCH1 D (CONCAT DIR1 D)
                                     T PROJECT)
                               (GIT-GET-FILE BRANCH2 D (CONCAT DIR2 D)
                                     T PROJECT)))
                    (DELETED ;; Shouldn't exist in BRANCH1, should exist in BRANCH2.  But maybe git is just confused in marking a file
                             ;; that exists in the wrong place as a delete instead of an add, or maybe it may think of a file that doesn't
                             ;; exist at all as having been deleted.  Try for both, but don't cause an error.
                             (SETQ D (CADR D))
                             (OR (GIT-GET-FILE BRANCH2 D (CONCAT DIR2 D)
                                       T PROJECT)
                                 (GIT-GET-FILE BRANCH1 D (CONCAT DIR1 D)
                                       T PROJECT)))
                    (CHANGED                                ; Should exist in both branches
                             (SETQ D (CADR D))
                             (GIT-GET-FILE BRANCH1 D (CONCAT DIR1 D)
                                   T PROJECT)
                             (GIT-GET-FILE BRANCH2 D (CONCAT DIR2 D)
                                   T PROJECT))
                    ((RENAMED COPIED)
                                 ;; These entries are from-to filename pairs.  If (CADDR) is 100, only need to fetch one, because it
                                 ;; presumably has disappeared in BRANCH2 and reappeared in BRANCH1. MAPPINGS is returned
                                 ;; so the connection can be reestablished higher up.
                         ;; If renamed and then changed, for now treat as unrelated adds and deletes:  put both files in the fromgit directory.
                         ;; Perhaps the mapping should still figure out how to relate them.

                         ;; For copied files, presumably 2 files are exactly the same.  But we hope we can show them on the same line, by
                         ;; virtue of the mapping.

                         [LET ((GFILE (CDR D))
                               F1 F1)

                              ;; GFILE is a triple (F2 F1 N )

                              ;; F1 is the file in branch 1, if any, F2 is in branch 2

                              (SETQ F1 (GIT-GET-FILE BRANCH1 (CADR GFILE)
                                             (CONCAT DIR1 (CADR GFILE))
                                             T PROJECT))
                              (SETQ F2 (GIT-GET-FILE BRANCH2 (CADR GFILE)
                                             (CONCAT DIR2 (CADR GFILE))
                                             T PROJECT))

                              ;; Let the directories figure it out

                              (AND NIL (if (EQ (CADDR GFILE)
                                               100)
                                           then

                                       ;; A little tricky to figure out what corresponds to the real file in the mapping, which directory it belongs
                                       ;; to.  Maybe the first one should always be one that exists, the second may just be a useful name.
                                       ;; But we have to know whether to match against INFO1 or INFO2

                                           (HELP GFILE 100)
                                           (push MAPPINGS (LIST (LIST)
                                                                (FULLNAME F1)
                                                                (SLASHIT (U-CASE (CONCAT DIR2
                                                                                         (CAR GFILE)))
                                                                         T)
                                                                (NTHCHAR (CAR D)
                                                                         1)
                                                                100))
                                           else ;; If not a perfect match, then the directory should figure it out
```

```
                                                (GIT-GET-FILE BRANCH2 (CAR GFILE)
                                                       (CONCAT DIR2 (CAR GFILE))
                                                       T PROJECT])
                                    (HELP "UNKNOWN GIT-DIFF TAG" D)))
                            (LIST DIR1 DIR2 MAPPINGS))])
```

(GIT-BRANCHES-COMPARE-DIRECTORIES

```
  [LAMBDA (BRANCH1 BRANCH2 LOCAL PROJECT)                    ; Edited 12-Jun-2024 22:52 by mth
                                                             ; Edited 10-Jun-2024 18:42 by mth
                                                             ; Edited  1-May-2024 14:58 by rmk
                                                             ; Edited 26-Sep-2023 22:40 by rmk
                                                             ; Edited 10-Jun-2023 17:28 by rmk
                                                             ; Edited 12-Sep-2022 14:41 by rmk
                                                             ; Edited 20-Jul-2022 21:18 by rmk
                                                             ; Edited 22-May-2022 22:47 by rmk
                                                             ; Edited  9-May-2022 15:14 by rmk
                                                             ; Edited  3-May-2022 23:04 by rmk
    (SETQ PROJECT (GIT-GET-PROJECT PROJECT))
    (SETQ BRANCH1 (if BRANCH1
                      then (GITORIGIN BRANCH1 LOCAL)
                    else (GIT-WHICH-BRANCH PROJECT T)))
    (LET (CDVALUE DIRS NENTRIES MAPPINGS (SHORT1 (GIT-SHORT-BRANCH-NAME BRANCH1))
                (SHORT2 (GIT-SHORT-BRANCH-NAME BRANCH2)))
         (PRINTOUT T "Comparing all " (L-CASE (fetch PROJECTNAME of PROJECT)
                                              T)
                " subdirectories of " SHORT1 " and " SHORT2 T)
         (PRINTOUT T "Fetching differences" T)
         (SETQ DIRS (GIT-GET-DIFFERENT-FILES BRANCH1 BRANCH2 NIL NIL PROJECT))
         (SETQ MAPPINGS (CADDR DIRS))
         (if DIRS
             then (TERPRI T)
                  ;; INCLUDEDFILES parameter to COMPAREDIRECTORIES needs to allow both top-level files, and leading dot filenames.
                  [SETQ CDVALUE (COMPAREDIRECTORIES (CAR DIRS)
                                                    (CADR DIRS)
                                                    '(> < ~= -* *-)
                                                    '(*.* *>*.* .* *>.*)
                                                    (GIT-GET-PROJECT PROJECT 'EXCLUSIONS)
                                                    NIL NIL NIL NIL (LIST (PACKFILENAME 'HOST NIL 'BODY (CAR DIRS))
                                                                          (PACKFILENAME 'HOST NIL 'BODY (CADR DIRS]
                  ;; We know that both sides come from Unix/unversioned, even if they have been copied into versioned FROMGIT, so we make a
                  ;; pass to remove the misleading versions.
                  ;;  Also, lower case and slash directories.  Perhaps can be done when the files are fetched?
                  [CDMAP CDVALUE
                         (FUNCTION (LAMBDA (CDE)
                                     (DECLARE (USEDFREE INFO1 INFO2))
                                     (LET [(MAP (CL:UNLESS INFO2
                                                    (find M in MAPPINGS suchthat (STRING.EQUAL (CAR M)
                                                                                               (fetch (CDINFO FULLNAME)
                                                                                                  of INFO1)
                                                                                               FILEDIRCASEARRAY)))]
                                         (CL:WHEN MAP
                                             (HELP 'MAP MAP))
                                         (CL:WHEN INFO1
                                             (change (fetch (CDINFO FULLNAME) of INFO1)
                                                     (SLASHIT (PACKFILENAME.STRING 'VERSION NIL 'BODY DATUM)
                                                              T)))
                                         (CL:WHEN INFO2
                                             (change (fetch (CDINFO FULLNAME) of INFO2)
                                                     (SLASHIT (PACKFILENAME.STRING 'VERSION NIL 'BODY DATUM)
                                                              T)))
                                         (if MAP
                                             then
                                         ;; This handles renames and copies.  We want the nominal source of a rename to be in the first
                                         ;; column, even though the target location is the one that was fetched.
                                                  (replace (CDENTRY INFO2) of CDE
                                                     with (create CDINFO
                                                                  FULLNAME _ (CADR MAP)
                                                                  DATE _ (CL:IF (EQ 'R (CADDR MAP))
                                                                                " <-"
                                                                                " ==")
                                                                  LENGTH _ ""
                                                                  AUTHOR _ ""
                                                                  TYPE _ ""
                                                                  EOL _ ""))
                                                  (replace (CDENTRY DATEREL) of CDE with (CADDR MAP]
                  (TERPRI T)
                  (if (fetch (CDVALUE CDENTRIES) of CDVALUE)
                      then (SETQ LAST-BRANCH-CDVALUE CDVALUE)
                           (CDBROWSER CDVALUE (CONCAT (L-CASE (fetch PROJECTNAME of PROJECT)
                                                              T)
                                                      " " SHORT1 "  vs  " SHORT2 "      "
                                                      (LENGTH (fetch (CDVALUE CDENTRIES) of CDVALUE))
                                                      " files")
```

```
                                (LIST SHORT1 SHORT2)
                                '(LABELFN GIT-CD-LABELFN BRANCH1 ,BRANCH1 BRANCH2 ,BRANCH2 PROJECT ,PROJECT)
                                GIT-CDBROWSER-SEPARATE-DIRECTIONS
                                '(Compare See))
                          (SETQ NENTRIES (LENGTH (fetch (CDVALUE CDENTRIES) of CDVALUE)))
                          (LIST NENTRIES (CL:IF (EQ NENTRIES 1)
                                                'difference
                                                'differences))
                    else '(0 differences))
          else '(0 differences])
```

(GIT-WORKING-COMPARE-DIRECTORIES

```
  [LAMBDA (SUBDIRS SELECT EXCLUDEDFILES FIXDIRECTORYDATES UPDATE PROJECT)
```

;; Edited 12-Jun-2024 22:52 by mth

;; Edited 26-Sep-2023 22:41 by rmk

;; Edited 17-Jun-2023 22:54 by rmk

;; Edited 10-Jun-2023 21:32 by rmk

;; Edited 20-Jul-2022 21:18 by rmk

;; Edited 25-Jun-2022 21:37 by rmk

;; Edited 17-May-2022 17:39 by rmk

;; Edited 10-May-2022 10:41 by rmk

;; Edited 29-Mar-2022 13:58 by rmk: working medley subdirectories with the current local git branch.

```
    (SETQ PROJECT (GIT-GET-PROJECT PROJECT))
    (CL:WHEN UPDATE (GIT-REMOTE-UPDATE NIL PROJECT))          ; Doesn't matter if we are looking only at local files in the current
                                                              ; branch.  We aren't fetching or checking out.
    (CL:UNLESS (AND (fetch GITHOST of PROJECT)
                    (fetch WHOST of PROJECT))
        (ERROR (fetch PROJECTNAME of PROJECT)
               " does not have both git and working directories"))
    (CL:WHEN (AND (LISTP SUBDIRS)
                  (NULL (CDR SUBDIRS)))
        (SETQ SUBDIRS (CAR SUBDIRS)))
    (CL:UNLESS SUBDIRS
        (SETQ SUBDIRS (OR (fetch DEFAULTSUBDIRS of PROJECT)
                          'ALL)))
    (SETQ SUBDIRS (L-CASE SUBDIRS))
    (LET ((SUBDIRSTRING (if (EQ SUBDIRS 'all)
                            then (SETQ SUBDIRS (ALLSUBDIRS PROJECT))
                                 "ALL subdirectories"
                            else SUBDIRS)))
         (for SUBDIR TITLE CDVAL (WPROJ _ (CONCAT "Working " (L-CASE (fetch PROJECTNAME of PROJECT)
                                                                     T)))
            (NENTRIES _ 0)
            (BRANCH2 _ (GIT-WHICH-BRANCH PROJECT T))
           first (PRINTOUT T "Comparing " SUBDIRSTRING 6 " of " WPROJ " and Git " BRANCH2 T)
                 (BKSYSBUF " ")
           inside SUBDIRS collect (TERPRI T)
                                  (SETQ CDVAL (COMPAREDIRECTORIES (MYMEDLEYSUBDIR SUBDIR T PROJECT)
                                                                  (GITSUBDIR SUBDIR T PROJECT)
                                                                  (OR SELECT '(> < ~= -* *-))
                                                                  NIL
                                                                  (for E DPOS in (GIT-GET-PROJECT PROJECT 'EXCLUSIONS)
                                                                     collect (SETQ DPOS (STRPOS SUBDIR (FILENAMEFIELD
                                                                                                         E
                                                                                                         'DIRECTORY)
                                                                                                1 NIL T T FILEDIRCASEARRAY))
                                                                             (CL:IF DPOS
                                                                                 (SUBSTRING E (ADD1 DPOS))
                                                                                 E))
                                                                  NIL NIL NIL FIXDIRECTORYDATES))
                                  [for CDE in (fetch CDENTRIES of CDVAL)
                                     do (CL:WHEN (fetch INFO1 of CDE)
                                                 (change (fetch (CDINFO FULLNAME) of (fetch INFO1 of CDE))
                                                         (UNSLASHIT DATUM T)))
                                        (CL:WHEN (fetch INFO2 of CDE)
                                                 (change (fetch (CDINFO FULLNAME) of (fetch INFO2 of CDE))
                                                         (SLASHIT DATUM T)))]
                                  CDVAL
           finally ;; Set up the browsers after everything has been done, otherwise if the user doesn't pay attention it might hang waiting for a
                   ;; region.
                   (CL:WHEN (AND (CDR $$VAL)
                                 GIT-MERGE-COMPARES)
                       (SETQ $$VAL (CDMERGE $$VAL))
                       [SETQ SUBDIRS (CONCATLIST (for SUBDIR in SUBDIRS collect (CONCAT SUBDIR "  "])
                   [for CDVAL TITLE in $$VAL as SUBDIR inside SUBDIRS
                      do (SETQ TITLE (CONCAT WPROJ "  vs.   " BRANCH2 "      " SUBDIR "          "
                                             (LENGTH (fetch (CDVALUE CDENTRIES) of CDVAL))
                                             " files"))
                         [CDBROWSER CDVAL TITLE '(,WPROJ ,BRANCH2)
                                '(BRANCH1 ,WPROJ BRANCH2 ,BRANCH2 SUBDIR ,SUBDIR LABELFN GIT-CD-LABELFN PROJECT
```

```
                                ,PROJECT)
                            GIT-CDBROWSER-SEPARATE-DIRECTIONS
                            `(Compare See "" Copy% <- (|Delete ALL <-| GIT-CD-MENUFN)
                                    ,@(CL:UNLESS (GIT-MAINBRANCH? BRANCH2 PROJECT T)
                                            '("" Copy% -> (Delete% -> GIT-CD-MENUFN)))]
                        (CONS (CONCAT SUBDIR "/")
                              (for CDENTRY in (fetch CDENTRIES of CDVAL) collect (fetch MATCHNAME of CDENTRY)))
                        (add NENTRIES (LENGTH (fetch (CDVALUE CDENTRIES) of CDVAL]
                    (SETQ LAST-WMEDLEY-CDVALUES $$VAL)
                    (TERPRI T)
                    (RETURN (LIST NENTRIES (CL:IF (EQ NENTRIES 1)
                                                  'difference
                                                  'differences)])
```

(GIT-COMPARE-WORKTREE

```
  [LAMBDA (BRANCH DONTUPDATE PROJECT)                    ; Edited  7-Jul-2022 11:17 by rmk
                                                         ; Edited  9-May-2022 16:17 by rmk

    (CL:UNLESS DONTUPDATE
        (GIT-ADD-WORKTREE BRANCH T PROJECT)
        (GIT-ADD-WORKTREE (GIT-MAINBRANCH PROJECT)
               T PROJECT))
    (LET (ADDEDFILES DELETEDFILES SOURCEFILES COMPILEDFILES OTHERFILES (MAINBRANCH (GIT-MAINBRANCH PROJECT)))
         (CL:UNLESS DONTUPDATE
             (GIT-ADD-WORKTREE BRANCH T PROJECT)
             (GIT-ADD-WORKTREE MAINBRANCH T PROJECT))
         (PRINTOUT T T "Comparing " (GIT-GET-PROJECT PROJECT 'PROJECTNAME)
                (FETCH PROJECTNAME OF PROJECT)
                " origin/" BRANCH " and " MAINBRANCH T)
         (FOR FILE BFILE MFILE IN (GIT-BRANCH-DIFF BRANCH MAINBRANCH PROJECT)
            DO (SETQ BFILE (INFILEP (CONCAT (WORKTREEDIR BRANCH PROJECT)
                                            FILE)))
               (SETQ MFILE (INFILEP (CONCAT (WORKTREEDIR MAINBRANCH PROJECT)
                                            FILE)))
               (IF (AND BFILE MFILE)
                   THEN (IF (NOT (LISPSOURCEFILEP BFILE))
                            THEN (PUSH OTHERFILES FILE)
                          ELSEIF (MEMB (U-CASE (FILENAMEFIELD BFILE 'EXTENSION))
                                       *COMPILED-EXTENSIONS*)
                            THEN (PUSH COMPILEDFILES FILE)
                            ELSE (PUSH SOURCEFILES FILE))
                 ELSEIF BFILE
                   THEN (PUSH ADDEDFILES FILE)
                   ELSE (PUSH DELETEDFILES FILE)))
         (CL:WHEN ADDEDFILES
             (PRINTOUT T T "Added files: " T)
             (FOR F IN (SORT ADDEDFILES) DO (PRINTOUT T 2 F T)))
         (CL:WHEN DELETEDFILES
             (PRINTOUT T T "Deleted files: " T)
             (FOR F IN (SORT ADDEDFILES) DO (PRINTOUT T 2 F T)))
         (CL:WHEN SOURCEFILES
             (PRINTOUT T T "Changed Medley source files:" T)
             (FOR FILETAIL FILE BFILE MFILE ON (SORT SOURCEFILES) DO (SETQ FILE (CAR FILETAIL))
                                                                     (PRINTOUT T 2 FILE T)
                                                                     (SETQ FILE (CAR FILETAIL))
                                                                     (SETQ BFILE
                                                                      (INFILEP (CONCAT (WORKTREEDIR BRANCH
                                                                                                    PROJECT)
                                                                                       FILE)))
                                                                     (SETQ MFILE
                                                                      (INFILEP (CONCAT (WORKTREEDIR
                                                                                                    MAINBRANCH
                                                                                                    PROJECT)
                                                                                       FILE)))
                                                                     (COMPARESOURCES-TEDIT BFILE MFILE)
                                                                     (TTY.PROCESS T)
                                                                     (CL:WHEN (OR OTHERFILES (CDR FILETAIL))
                                                                            (WAITFORINPUT))))
         (CL:WHEN COMPILEDFILES
             (PRINTOUT T T "Medley compiled files, no comparisons:")
             (FOR F IN COMPILEDFILES DO (PRINTOUT T 2 F T)))
         (CL:WHEN OTHERFILES
             (PRINTOUT T T "Other changed files, using TEDIT-SEE")
             (FOR FILETAIL FILE BFILE MFILE ON (SORT OTHERFILES) DO (SETQ FILE (CAR FILETAIL))
                                                                    (PRINTOUT T 2 FILE)
                                                                    (SETQ FILE (CAR FILETAIL))
                                                                    (SETQ BFILE (INFILEP (CONCAT (WORKTREEDIR
                                                                                                  BRANCH
                                                                                                  PROJECT)
                                                                                                 FILE)))
                                                                    (SETQ MFILE (INFILEP (CONCAT (WORKTREEDIR
                                                                                                  MAINBRANCH
                                                                                                  PROJECT)
                                                                                                 FILE)))
                                                                    (COMPARETEXT BFILE MFILE 'LINE)
                                                                    (AND NIL (TEDIT-SEE BFILE)
                                                                         (TEDIT-SEE MFILE))
```

```
                                                                (TTY.PROCESS T)
                                                                (CL:WHEN (CDR FILETAIL)
                                                                       (WAITFORINPUT))))])
```

(GITCDOBJBUTTONFN

```
[LAMBDA (OBJ WINDOW)                                                    ; Edited 10-May-2022 00:30 by rmk
  (HELP)
  (LET
   ([CDENTRY (CAR (IMAGEOBJPROP OBJ 'OBJECTDATUM]
    (BRANCH1 (WINDOWPROP WINDOW 'BRANCH1))
    (FONT (FONTCREATE 'TERMINAL 10))
    COPYITEM COMPAREITEMS TYPE INFO1 INFO2)
   (CL:WHEN (AND CDENTRY (CADR (IMAGEOBJPROP OBJ 'OBJECTDATUM))
                 (EQ LASTKEYBOARD 0))
       (SETQ INFO1 (FETCH (CDENTRY INFO1) OF CDENTRY))
       (SETQ INFO2 (FETCH (CDENTRY INFO2) OF CDENTRY))
       [IF (MOUSESTATE (ONLY LEFT))
           THEN [SETQ COMPAREITEMS
                      (IF (AND INFO1 INFO2)
                          THEN [IF (EQ (SETQ TYPE (FETCH (CDINFO TYPE) OF INFO1))
                                       (FETCH (CDINFO TYPE) OF INFO2))
                                   THEN (SELECTQ TYPE
                                                 (SOURCE [LIST (LIST "Compare sources?" ''COMPARESOURCES)
                                                               (LIST "Examine sources?" ''EXAMINE])
                                                 (COMPILED)
                                                 (TEXT (LIST (CONCAT "Compare text files?")
                                                             ''TEXT))
                                                 (IF (MEMB (U-CASE (FILENAMEFIELD (FETCH (CDINFO FULLNAME)
                                                                                         OF INFO1)))
                                                           '(TEXT TXT))
                                                     THEN [LIST (LIST "Compare text files?" (KWOTE TYPE)
                                                                      ''COMPARETEXT]
                                                   ELSE (LIST (LIST (CONCAT "See " TYPE " files?")
                                                                    (KWOTE TYPE]
                                                 NIL]
                                   ELSE (LIST (LIST "Show file?" ''TEDIT]
                        ELSEIF (OR INFO1 INFO2)
                          THEN (LIST (LIST "Show file?" ''TEDIT]
         ELSEIF [AND (MOUSESTATE (ONLY MIDDLE))
                     (NOT (WINDOWPROP WINDOW 'READONLY]
           THEN (SETQ COPYITEM (CONS (SELECTQ (CADDR (IMAGEOBJPROP OBJ 'OBJECTDATUM))
                                              (LEFT (LIST (CONCAT "Copy TO git " (GIT-WHICH-BRANCH)
                                                                  "?")
                                                          ''TOGIT))
                                              (RIGHT (LIST (CONCAT "Copy FROM git " (GIT-WHICH-BRANCH)
                                                                   "?")
                                                           ''FROMGIT))
                                              NIL]
       (CL:WHEN (OR COPYITEM COMPAREITEMS)
           (SELECTQ (MENU (CREATE MENU
                                  TITLE _ (CONCAT (WINDOWPROP WINDOW 'SUBDIR)
                                                  "/"
                                                  (FETCH MATCHNAME OF CDENTRY))
                                  ITEMS _ (APPEND COPYITEM COMPAREITEMS)
                                  MENUFONT _ FONT
                                  MENUTITLEFONT _ FONT))
                    (TOGIT (CL:WHEN (TOGIT (FETCH (CDINFO FULLNAME) OF INFO1)
                                           WINDOW)
                                    (IMAGEOBJPROP OBJ 'COPIED T)
                                    (REDISPLAYW WINDOW)
                                    (CDOBJ.DISPLAYFN OBJ WINDOW)))
                    (FROMGIT (CL:WHEN (FROMGIT (FETCH (CDINFO FULLNAME) OF INFO2)
                                               WINDOW)
                                      (IMAGEOBJPROP OBJ 'COPIED T)
                                      (AND NIL (REDISPLAYW WINDOW))))
                    (COMPARESOURCES
                        (TTY.PROCESS T)
                        (CSBROWSER (fetch (CDINFO FULLNAME) OF INFO1)
                                   (fetch (CDINFO FULLNAME) OF INFO2)))
                    (COMPARETEXT (TTY.PROCESS T)
                                 (COMPARETEXT (FETCH (CDINFO FULLNAME) OF INFO1)
                                              (FETCH (CDINFO FULLNAME) OF INFO2)
                                              'PARA))
                    (TEDIT (CL:WHEN INFO1
                                    (TEDIT-SEE (FETCH (CDINFO FULLNAME) OF INFO1)))
                           (CL:WHEN INFO2
                                    (TEDIT-SEE (FETCH (CDINFO FULLNAME) OF INFO2))))
                    NIL)))])
```

(GIT-CD-LABELFN

```
[LAMBDA (FILE1 FILE2 USERDATA)                                          ; Edited  5-Jan-2022 15:10 by rmk
                                                                        ; Edited 16-Dec-2021 12:25 by rmk
                                                                        ; Edited 13-Dec-2021 22:13 by rmk
  (DECLARE (USEDFREE CDVALUE))
  (LET (NC B LABEL1 LABEL2)
       (CL:WHEN (SETQ NC (FETCH NCDIR OF (FETCH CDMAXNC1 OF CDVALUE)))
           (SETQ LABEL1 (SLASHIT (SUBSTRING FILE1 (ADD1 NC))
```

```
                                T))
            (CL:WHEN (SETQ B (LISTGET USERDATA 'BRANCH1))
                (SETQ LABEL1 (CONCAT B "/" LABEL1))))
        (CL:WHEN (SETQ NC (FETCH NCDIR OF (FETCH CDMAXNC2 OF CDVALUE)))
            (SETQ LABEL2 (SLASHIT (SUBSTRING FILE2 (ADD1 NC))
                                T))
            (CL:WHEN (SETQ B (LISTGET USERDATA 'BRANCH2))
                (SETQ LABEL2 (CONCAT B "/" LABEL2))))
        (LIST (OR LABEL1 FILE1)
              (OR LABEL2 FILE2])
```

(GIT-CD-MENUFN

```
[LAMBDA (TBITEM MENUITEM CDBROWSER KEY)                    ; Edited 21-Sep-2022 21:34 by rmk
                                                           ; Edited 22-May-2022 19:13 by rmk
                                                           ; Edited  8-May-2022 09:26 by rmk
                                                           ; Edited 10-Dec-2021 08:52 by rmk
```

;; MENUITEM is of the form (display-atom <this function> . extrainfo). The selector for the selectq is either the CAR of the extrainfo or the display ;; atom

```
(DECLARE (USEDFREE FILE1 FILE2 LABEL2 TYPE CDENTRY))
(SELECTQ (OR (CADDR MENUITEM)
             (CAR MENUITEM))
     (Delete% -> (FLASHWINDOW PWINDOW)
                 (GIVE.TTY.PROCESS PWINDOW)
                 (CL:WHEN [OR (EQ KEY 'MIDDLE)
                              (EQ 'Y (ASKUSER NIL 'N (CONCAT "Delete " LABEL2 " ? "]
                     (GIT-DELETE-FILE FILE2 (LISTGET USERDATA 'PROJECT))
                     (TB.DELETE.ITEM CDBROWSER TBITEM)))
     (|Delete ALL <-|
          (FLASHWINDOW PWINDOW)
          (GIVE.TTY.PROCESS PWINDOW)
          (if (NAMEFIELD LABEL1 T)
              then (CL:WHEN [OR (EQ KEY 'MIDDLE)
                                (EQ 'Y (ASKUSER NIL 'N (CONCAT "Delete ALL versions of " (NAMEFIELD LABEL1 T)
                                                               " ? "]
                           (MYMEDLEY-DELETE-FILES FILE1 (LISTGET USERDATA 'PROJECT))
                           (TB.DELETE.ITEM CDBROWSER TBITEM))
            else (PRINTOUT T "Nothing to delete")))
     (Delete% BOTH (FLASHWINDOW PWINDOW)
                   (GIVE.TTY.PROCESS PWINDOW)
                   (CL:WHEN (EQ 'Y (ASKUSER NIL 'N (CONCAT "Delete all Medley and git versions of "
                                                           (NAMEFIELD LABEL1 T)
                                                           " ? ")))
                       (GIT-DELETE-FILE FILE2 (LISTGET USERDATA 'PROJECT))
                       (MYMEDLEY-DELETE-FILES FILE1 (LISTGET USERDATA 'PROJECT))
                       (TB.DELETE.ITEM CDBROWSER TBITEM)))
     (SHOULDNT])
```

(GIT-WORKING-COMPARE-FILES

```
[LAMBDA (FILE PROJECT)                                     ; Edited  7-Jul-2022 11:17 by rmk
                                                           ; Edited 22-May-2022 14:45 by rmk
  (LET ((FILE1 (UNSLASHIT (PACKFILENAME 'HOST (GIT-GET-PROJECT PROJECT 'WHOST)
                                'BODY FILE)
                       T))
        (FILE2 (SLASHIT (PACKFILENAME 'HOST (GIT-GET-PROJECT PROJECT 'GITHOST)
                              'BODY FILE)
                       T)))
       (CD-COMPARE-FILES FILE1 FILE2 FILE1 FILE2])
```

(GIT-BRANCHES-COMPARE-FILES

```
[LAMBDA (FILE BRANCH1 BRANCH2 PROJECT LOCAL)               ; Edited 22-May-2022 22:50 by rmk
  (SETQ PROJECT (GIT-GET-PROJECT PROJECT))
  (SETQ BRANCH1 (SELECTQ (U-CASE BRANCH1)
                     ((NIL T)
                          (GIT-MY-CURRENT-BRANCH PROJECT))
                     ((LOCAL REMOTE ORIGIN)
                          (GIT-PICK-BRANCH (GIT-BRANCHES BRANCH1 PROJECT T)))
                     BRANCH1))
  (SETQ BRANCH2 (SELECTQ (U-CASE BRANCH2)
                     ((NIL T)
                          (GIT-MAINBRANCH PROJECT LOCAL))
                     ((LOCAL REMOTE ORIGIN)
                          (GIT-PICK-BRANCH (GIT-BRANCHES BRANCH2 PROJECT T)))
                     BRANCH2))
  (LET ((FILE1 (GIT-GET-FILE BRANCH1 FILE NIL NIL PROJECT))
        (FILE2 (GIT-GET-FILE BRANCH2 FILE NIL NIL PROJECT)))
       (CD-COMPARE-FILES FILE1 FILE2 (CONCAT (GIT-SHORT-BRANCH-NAME BRANCH1)
                                             " " FILE)
              (CONCAT (GIT-SHORT-BRANCH-NAME BRANCH2)
                      " " FILE])
```

(GIT-PR-COMPARE

```
[LAMBDA (RB PROJECT)                                       ; Edited  6-Jul-2023 22:22 by rmk
```

```
    (GIT-BRANCHES-COMPARE-DIRECTORIES (GIT-MAINBRANCH PROJECT)
           RB NIL PROJECT])

)

(RPAQ? FROMGITN 0)

;;
;; Utilities

(DEFINEQ
```

(CDGITDIR

```
  [LAMBDA (PROJECT)                                                    ; Edited 23-Sep-2023 13:01 by rmk
                                                                       ; Edited  8-Jul-2022 10:34 by rmk
                                                                       ; Edited  7-Jul-2022 09:36 by rmk
                                                                       ; Edited  7-May-2022 22:41 by rmk
                                                                       ; Edited  2-Nov-2021 21:12 by rmk:
    (CONCAT "cd " (SLASHIT (TRUEFILENAME (fetch GITHOST of PROJECT))
                           NIL T)
           " && "])
```

(GIT-COMMAND

```
  [LAMBDA (CMD ALL NOERROR PROJECT)                                    ; Edited 16-Jul-2022 13:06 by rmk
                                                                       ; Edited  8-Jul-2022 10:20 by rmk
                                                                       ; Edited  7-May-2022 22:40 by rmk
                                                                       ; Edited  7-Oct-2021 11:15 by rmk:
    ;; Suppress .git lines unless ALL
    (SETQ PROJECT (GIT-GET-PROJECT PROJECT))
    (CL:UNLESS (OR (EQ 1 (STRPOS "git" CMD))
                   (EQ 1 (STRPOS "gh" CMD)))
        (SETQ CMD (CONCAT "git " CMD)))
    [bind LPOS while (SETQ LPOS (STRPOS "local/" CMD)) do (SETQ CMD (CONCAT (SUBSTRING CMD 1 (SUB1 LPOS))
                                                                             (SUBSTRING CMD (IPLUS LPOS
                                                                                                   (NCHARS
                                                                                                          "local/"]
    (LET (LINES (RESULTFILE (GIT-COMMAND-TO-FILE CMD PROJECT NOERROR)))
         (CL:WHEN (LISTP RESULTFILE)                                   ; CADR is Unix error stream
             (CL:WITH-OPEN-FILE (ESTREAM (CADR RESULTFILE)
                                         :DIRECTION :INPUT :EXTERNAL-FORMAT (SYSTEM-EXTERNALFORMAT))
                    (COPYCHARS ESTREAM T))
             (DELFILE (CADR RESULTFILE))
             (SETQ RESULTFILE (CAR RESULTFILE)))
         (CL:WHEN RESULTFILE
             (SETQ LINES (GIT-RESULT-TO-LINES RESULTFILE ALL))
             (DELFILE RESULTFILE)                                      ; On tmp/, OK if we miss
            LINES)])
```

(GITORIGIN

```
  [LAMBDA (BRANCH LOCAL)                                               ; Edited  9-May-2022 14:26 by rmk
                                                                       ; Edited 25-Nov-2021 08:47 by rmk:
                                                                       ; Edited 22-Nov-2021 17:29 by rmk:
    ;; Insures origin/ unless LOCAL or local/ already
    (CL:UNLESS BRANCH (HELP "BRANCH MUST BE SPECIFIED"))
    (if (OR (STRPOS "origin/" BRANCH)
            (STRPOS "local/" BRANCH))
        then BRANCH
      else (CONCAT (CL:IF LOCAL
                          "local/"
                          "origin/")
                   BRANCH])
```

(GIT-INITIALS

```
  [LAMBDA NIL                                                          ; Edited 19-Jan-2022 13:18 by rmk
    (OR (CL:IF (EQ (CHARCODE %:)
                   (NTHCHARCODE INITIALS -1))
               (SUBSTRING INITIALS 1 -2)
               INITIALS)
        (ERROR "INITIALS is not set"])
```

(GIT-COMMAND-TO-FILE

```
  [LAMBDA (CMD PROJECT NOERROR)                                        ; Edited 18-Jul-2022 09:53 by rmk
                                                                       ; Edited 16-Jul-2022 10:09 by rmk
                                                                       ; Edited  9-Jul-2022 18:55 by rmk
                                                                       ; Edited  8-Jul-2022 08:51 by rmk
    ;; Try to make the temporary name unique.  Maybe Unix mktemp, except that we need to know the name that was used.  So we calculate it,
    ;; provide it, and assume that it worked.  Caller an decide to delete it after examination.  (Or, left to be reaped from /tmp/)
    (SETQ PROJECT (GIT-GET-PROJECT PROJECT))
    ;; Filename of  the form /tmp/medley-gitresult-{IDATE}-{rand}
```

```
    (SETQ CMD (STRIPLOCAL CMD))
    (LET* ([PROJECTNAME (L-CASE (GIT-GET-PROJECT PROJECT 'PROJECTNAME]
           (DATE (IDATE))
           (RAND (RAND))
           (RESULTFILE (CONCAT "{UNIX}/tmp/" PROJECTNAME "-" DATE "-" RAND "-result"))
           (ERRORFILE (CONCAT "{UNIX}/tmp/" PROJECTNAME "-" DATE "-" RAND "-error"))
           COMPLETIONCODE)
          [SETQ COMPLETIONCODE (PROCESS-COMMAND (CONCAT (CDGITDIR PROJECT)
                                                        CMD " > " (STRIPHOST RESULTFILE)
                                                        " 2> "
                                                        (STRIPHOST ERRORFILE]
          (CLOSEF? ERRORFILE)
          (CLOSEF? RESULTFILE)
          (CL:WHEN [AND (INFILEP ERRORFILE)
                        (IEQP 0 (GETFILEINFO ERRORFILE 'LENGTH]
              (DELFILE ERRORFILE)
              (SETQ ERRORFILE NIL))
          (CL:WHEN (AND (INFILEP RESULTFILE)
                        (IEQP 0 (GETFILEINFO RESULTFILE 'LENGTH))
                       ERRORFILE)
              (DELFILE RESULTFILE)                          ; Don't delete if the error file is also empty
              (SETQ RESULTFILE NIL))
          (CL:WHEN (AND (EQ COMPLETIONCODE 0)
                        ERRORFILE)                          ; Check the error file, just in case
              (CL:WITH-OPEN-FILE (ESTREAM ERRORFILE :DIRECTION :INPUT :EXTERNAL-FORMAT (SYSTEM-EXTERNALFORMAT))
                     (CL:WHEN (OR (EQ 0 (OR (FILEPOS "fatal: " ESTREAM 0 1)
                                            (FILEPOS "gh: Command not found" ESTREAM 0 1)
                                            (FILEPOS "unknown command %"" ESTREAM 0 1)))
                                  (FILEPOS "' is not a git command." ESTREAM (NCHARS CMD)))
                              (SETQ COMPLETIONCODE 1))))
          (if (EQ 0 COMPLETIONCODE)
              then (if (AND RESULTFILE ERRORFILE)
                       then (LIST RESULTFILE ERRORFILE)
                     elseif RESULTFILE
                     else ERRORFILE)
            else (DELFILE RESULTFILE)
                 (DELFILE ERRORFILE)
                 (CL:UNLESS NOERROR
                     (ERROR (CONCAT "Command failed: " CMD)))
                 NIL])
```

(GIT-RESULT-TO-LINES

```
  [LAMBDA (FILE ALL)                                        ; Edited 16-Jul-2022 22:21 by rmk
    ;; Suppress .git lines unless ALL
    (CL:WITH-OPEN-FILE (STREAM FILE :DIRECTION :INPUT :EXTERNAL-FORMAT (SYSTEM-EXTERNALFORMAT))
           (bind LINE until (EOFP STREAM) when [PROGN (SETQ LINE (CL:READ-LINE STREAM :EOF-ERROR-P NIL :EOF-VALUE
                                                                                     NIL))
                                                      (OR ALL (NOT (STRPOS ".git" LINE 1]
              collect LINE])
```

(STRIPLOCAL

```
  [LAMBDA (STRING)                                          ; Edited 18-Jul-2022 09:52 by rmk
    ;; Removes local/ substrings wherever they appear.  To be used in coerecing from a lisp internal convention that local branches carry a local tag to
    ;; the git convention that an unqualified name is local.
    [bind POS while (SETQ POS (STRPOS "local/" STRING))
       do (SETQ STRING (CONCAT (SUBSTRING STRING 1 (SUB1 POS))
                               (OR (SUBSTRING STRING (IPLUS POS (CONSTANT (NCHARS "local/")))
                                              -1)
                                   ""]
    STRING])

)

(PUTPROPS GITFNS FILETYPE :TCOMPL)
```

FUNCTION INDEX

VARIABLE INDEX

COMMAND INDEX

RECORD INDEX

PROPERTY INDEX