```
1
2   # pet.awk - version 1.20 #
3
4   ##############################################################################
5   #
6   #
7   # PROGRAM:        pet.awk
8   #
9   # Programmer:     Garry Daly
10  #
11  # Company:        TBD
12  #
13  # Created:        Fri Sep 22
14  #
15  # Abstract:       This program will log messages to rop for package testing
16  #
17  # Description:    This program will log session start and stop times to
18  #                 the rop, along with the id testing start and stop times
19  #
20  # Contents:
21  #
22  # Modified:
23  #
24  #   DATE    DEV                          CHANGE ABSTRACT
25  # ........  ...   ..........................................................
26  # 09/22/xx  grd   program birthday -- its a clone ...
27  # 10/04/xx  grd   0.1.0  - available
28  # 10/13/xx  grd   0.1.1  - allow id overlapping and idcount
29  # 10/17/xx  grd   0.1.2  - speed up - do logging in background
30  # 10/26/xx  grd   0.1.3  - add previous package file, allow input after
31  #                          package test time, initialize opnid string
32  #                          upon start up (i.e. users leaving ids open,
33  #                          quitting then re-executing pet)
34  # 10/27/xx  grd   0.1.4  - added labid for nsc4 and lab y testing.
35  # 11/16/xx  grd   0.1.5  - added labid for v2 and allowed PHONE numbers
36  #                          like (PHONE=IXxxxxx|xxxxx|nnn-xxx-xxxx)
37  # 01/19/xx  grd   0.1.6  - added id for b, DONTASK and input of package name
38  # 06/08/xx  grd   0.1.7  - added ability to write comments to the rop
39  #                          added labid database for support
```

```
#                           GOAT item 1139.
# 06/19/xx  grd   0.1.8  - added more checking for new/old tn's and simplified
#                           code.
#                           GOAT item 1139.
# 06/30/xx  grd   1.1    - modified for and placed under sccs
# 08/02/xx  grd          - changed 708 area code to 630
# 10/02/xx  grd   1.3    - post changed new handles for name@url.com
#                           changed to use the old handle login!machine
# 10/08/xx  rem   1.4    - added call load tools menu to the main menu, this
#                           will allow users to stop & start ACS and
#                  kill 2stpe, more functionality will be added later.
#
# 12/05/xx  grd   added remote pet access.  Create database files to insert
# 12/20/xx        or update the official database on a remote machine.
#                 Created a seperate function to display the MAIN menu
#                 headings.  Redesigned menu mechanism.  Bug fixed updating
#                 id for a different labid.
#                 Turned OFF call load tools.
#                 GOAT item 1601.
#
# 12/05/xx  grd   added user logging capability so that known lab problems
# 12/20/xx        can be inputted and viewed by the user population.  Added
#                 PAGER and EDITOR varables.
#                 GOAT item 1140.
# 01/05/xx  grd   modified tpfx to genrate a more unique file name
#                 GOAT item 1612.
# 01/22/xx  grd   modified so that uucp is not done.  Files are now kept
#                 in the pet_xfer directory.
#                 GOAT item 1646.
# 01/30/xx  grd   add id list capability for pet.  This allows a list of
#                 ids to begin and end.
#                 GOAT item 1666.
# 02/04/xx  grd   bug file file generation for GOAT item 1666
# 05/30/xx  grd   someone changed the craft shell so that the ":::" is no
#                 longer allowed as an argument, as in the following example:
#
#                      exc:ENVIR:uproc,fn="/unixa/users/prtmsg",       \
#                      args="::: FINISH  :::  mike h weis  x3000        \
#                      ID: 590942";                                      \
```

```
 #                    ?D - MISSING DATA - DATA BLOCK - THIRD KEYWORD
 #
 #                I don't who changed it or why they did, but I corrected
 #                the problem by makeing the colons, plus signs.
 #                GOAT item 1773.
 # 12/11/xx  grd  bug fix writing to the rop
 # 12/16/xx  grd  moved tempnames() call inside id admin to insure each
 #                time an id is inserted or updated, unique file names
 #                can be generated for remote pet users.
 #
 ##############################################################################/

#awkcchint -o pet

BEGIN {
     Version  = "1.20";
     Delta    = "12/16/97 - 23:45:17";

     Provider = "John Smith - jsmith@aol.com - x92000"

     TCMD     = "tput clear";


     ###########################################################
     #  Run in debug mode if were only debugging this tool.  #
     #  Both DEBUG and DLEVEL must be exported.              #
     #                                                       #
     #  DEBUG  must be set to ON | OFF                       #
     #  DLEVEL must be set to 0  | > = 1 < = 99             #
     #                                                       #
     ###########################################################

     cmd = "env | grep \"^DEBUG=\"";
     cmd | getline dflag;
     close(cmd);

     debug = "";
     if (dflag != "")
     {
```

```
118         #  Remove the 'DEBUG=' part
119
120         split(dflag, a, "=");
121         debug = a[2];
122     }
123     else
124         debug = "OFF";
125
126     if ((debug == "on") || (debug == "ON"))
127         debug = "ON";
128     else
129         debug = "OFF";
130
131     cmd = "env | grep \"^DLEVEL=\"";
132     cmd | getline lflag;
133     close(cmd);
134
135     dlevel = "";
136     if (lflag != "")
137     {
138         #  Remove the 'DLEVEL=' part
139
140         split(lflag, a, "=");
141         dlevel = a[2];
142         if (dlevel < 0)
143             dlevel = 0;
144         else
145             if (dlevel > 99)
146                 dlevel = 99;
147     }
148     else
149          dlevel = 0;
150
151     #
152     #  Both debug must equal ON and dlevel > 0 for debugging
153     #
154
155     if ((debug == "ON") && (dlevel > 0))
156         debug = "ON";
```

```
157         else
158             debug = "OFF";
159
160         ltslogin = "surat";
161         ltsmach  = "lts08";
162
163         cmd = "logdir exptools " ltslogin;
164         cmd | getline tools;
165         cmd | getline home;
166         close(cmd);
167
168
169         curpkg  = home "/reglib/current/bwmno";
170         prepkg  = home "/reglib/previous/bwmno";
171         rmsuid  = home "/regbin/rmsuid";
172
173         if (debug == "ON")
174         {
175             idb     = home "/reglib/Tdb/idb";
176             tdb     = home "/reglib/Tdb/testerdb";
177             tidx    = home "/reglib/Tdb/tstindx";
178         }
179         else
180         {
181             idb     = home "/reglib/dbase/idb";
182             tdb     = home "/reglib/dbase/testerdb";
183             tidx    = home "/reglib/dbase/tstindx";
184         }
185
186         ldb     = home "/reglib/dbase/ldb";
187         lprobdb = home "/reglib/current/lprobdb";
188
189         Didb    = home "/reglib/define/idb";
190         Dtdb    = home "/reglib/define/testerdb";
191         Dldb    = home "/reglib/define/ldb";
192         Dtidx   = home "/reglib/define/tstindx";
193
194         UPDATE  = tools "/lib/unity/bin/alter -q -I";
195         INSERT  = tools "/lib/unity/bin/insert -q -I";
```

```
196         QUERY   = tools "/lib/unity/bin/uselect -q -I";
197
198         dash38  = "--------------------------------------";
199         mmlcmd  = "\"exc:ENVIR:uproc,fn=\\\"/unixa/users/prtmsg\\\""
200         mmlcmd  = mmlcmd ",args=\\\""
201
202         datecmd = "date +%m/%d/%y";
203         timecmd = "date +%H:%M:%S";
204         secscmd = "date +%S";
205
206         nodate  = "../../..";
207         notime  = "..:..:..";
208
209         package = "";
210         id     = "";
211         tindex  = "";
212         tname   = "";
213
214         OFS = FS = "|";
215
216         #
217         #  EXECUTABLE CODE STARTS HERE
218         #
219
220         if (debug == "ON")
221         {
222             #
223             #  Copy the official data base and only work on the copied version
224             #
225
226             cmd = "cp " home "/reglib/dbase/*" " " home "/reglib/Tdb/.";
227             system(cmd);
228             close(cmd);
229         }
230
231         next_state = "continue";
232         namechg    = "true";
233
234         #
```

```
235         #  Generate the temporary filenames
236         #
237
238         tempnames();
239
240         ##########################################
241         #  Determine the lab you are working with  #
242         ##########################################
243
244         shellvar = "LID";
245         getdollar();
246
247         suvlab = "yes";
248
249         if (vardef == "")
250         {
251                 cmd = "tbname | head -1";
252                 cmd | getline slabid;
253                 close(cmd);
254
255                 split(slabid, a, "-");
256                 labid = a[1];
257
258                 printf("\n□\n");
259                 printf("NOTICE:  No LID was exported!\n");
260                 printf("         Assuming  Testing in L %s\n", a[1]);
261                 printf("□\n");
262
263         }
264         else
265              labid = vardef;
266
267         #
268         #  Map the users lid to lname
269         #
270
271         record = "";
272
273         cmd = QUERY Dldb " from " ldb " where lid leq " labid;
```

```
274         cmd | getline record;
275         close(cmd);
276
277         if (record != "")
278         {
279             split(record, a, "|");
280              labstr = a[5];
281             suvlab = a[6];
282             nfslab = a[7];
283         }
284         else
285         {
286             cmd = QUERY Dldb " from " ldb " where lname leq " labid;
287             cmd | getline record;
288             close(cmd);
289
290             if (record != "")
291             {
292                 split(record, a, "|");
293                  labstr = a[5];
294                 suvlab = a[6];
295                 nfslab = a[7];
296             }
297             else
298             {
299                 printf("\n□\n");
300                 printf("ERROR:  Unknown LID: %s was exported!\n", rlabid);
301                 printf("        Exiting ... \n");
302                 printf("□\n");
303
304                 exit(1);
305
306             }
307         }
308
309         system(TCMD);
310         close(TCMD);
311
312         disptinfo();
```

```
313         getpkgno();
314
315         if (next_state == "exit")
316         {
317             thankyou();
318             exit(1);
319         }
320
321         ##############################################
322         #  Determine if the user is an expert user  #
323         ##############################################
324
325         shellvar = "DONTASK";
326         getdollar();
327
328         dontask = "false"
329         if (vardef != "")
330         {
331             if ((vardef == "y")   || (vardef == "Y")  ||
332                 (vardef == "yes") || (vardef == "YES") )
333
334                 dontask = "true";
335             else
336                 dontask = "false";
337         }
338
339         ###################################################################
340         #  If the user exported the PHONE variable - use it as the default  #
341         ###################################################################
342
343         shellvar = "PHONE";
344         getdollar();
345
346         if (vardef != "")
347         {
348             tn = vardef;
349
350             valtn();
351             if (tnstat == "ng")
```

```
            tn = "";
        else
        {
            #
            #  We got an exported PHONE number -- get the post address
            #  and select the tester index for the user
            #

            getpost();
        }
    }


    while (tn == "")
        getusrin();


    ###########################################################################
    #  Initialize the open ids string if the user dropped out of pet #
    #  and had open ids under test.                                    #
    ###########################################################################

    initopen();

    if (next_state == "exit")
        exit(1);

    ############################
    #  DISPLAY THE MAIN MENU #
    ############################

    mainmenu();

    thankyou();
}


function getdollar()
{
```

```
391       #
392       #  This function expands $xxx variables
393       #
394
395       var = "";
396
397       cmd = "env | grep \"^" shellvar "=\"";
398       cmd | getline var;
399       close(cmd);
400
401       if (var != "")
402       {
403           #  Remove the 'xxx= part
404
405           split(var, dollar, "=");
406           vardef = dollar[2];
407       }
408       else
409            vardef = "";
410
411  }
412
413  function gettilde()
414  {
415       #
416       #  This function expands ~xxx variables
417       #
418
419       vardef = "";
420
421       cmd = "echo `logdir " shellvar "` 2>/dev/null";
422       cmd | getline vardef;
423       close(cmd);
424  }
425
426  function topdisp()
427  {
428           #
429           #  This function displays the top portion of the menu's
```

```
    #
    # This function consumes 7 lines.
# 1 # ------------------------------------------------------------------
# 2 # LAST USER:   last,first middle     PACKAGE NUMBER:   YR-xxxx
# 3 # TEST LAB:    lab zzzzz             ACTIVE ID COUNT: 0
# 4 # ACTIVE IDS:
# 5 #
# 6 # ------------------------------------------------------------------
# 7 #                   PACKAGE ENVIRONMENT TEST TOOL


    #  Adjust the opnid string so that a maximum of 63 characters
    #  on the FIRST display line.  Hopfully no tester will do more
    #  then 15 ids at a time.

    if (length(opnid) > 63)
    {
       lastitem = 0;
       idline1 = "";
       idline2 = "";

       openitems = split(opnid, b, " ");
       for (j=1; j <= openitems; j++)
       {
           if ((length(idline1) + length(b[j])) > 62)
               break;
           else
           {
               if (idline1 == "")
                   idline1 = b[j];
               else
                   idline1 = idline1 " " b[j];
           }
           lastitem = j;
       }
       if (lastitem < openitems)
           for (j=lastitem + 1; j <= openitems; j++)
```

```
                {
                    if (idline2 == "")
                        idline2 = b[j];
                    else
                        idline2 = idline2 " " b[j];
                }
        }
        else
        {
           idline1 = opnid;
           idline2 = " ";
        }

        if (debug == "ON")
            printf("\n\n");
        else
        {
            system(TCMD);
            close(TCMD);
        }

        printf("  %s%s\n", dash38, dash38);

        printf("  LAST USER:   %24s%11s", tname, " ");
        printf("PACKAGE NUMBER:   %s \n", package);

        printf("  TEST LOC:    %-12s%s", lstr, "       ");
        printf("%15s ACTIVE ID COUNT: %d\n", " ", idcnt);
        printf("  ACTIVE IDS: %s\n", idline1);
        printf("              %s\n", idline2);

        printf("  %s%s\n", dash38, dash38);

        if (debug == "ON")
            printf("             PACKAGE ENVIRONMENT TEST TOOL - (TEST MODE VERSION)\n");
        else
            printf("                          PACKAGE ENVIRONMENT TEST TOOL\n");
}
```

```
508  function mainmenu()
509  {
510       next_state = "continue";
511       while (1)
512       {
513           ################################################
514           #  Loop continuously through the main screen  #
515           ################################################
516
517           if (next_state == "exit")
518               return;
519
520
521           #
522           #  Display the menu header
523           #
524
525           topdisp();
526
527           #
528           #  Lines 8-9  MENU NAME
529           #
530
531           printf("                                    MAIN MENU\n\n");
532
533           #  Line 10
534           printf("\t1) ID ADMIN        - logs id start/stop  time to the rop\n");
535           #  Line 11
536           printf("\t2) CHANGE USER      - changes user of the pet tool\n");
537
538           #  Line 12
539           printf("\t3) SPECIAL OPTIONS  - display special pet options\n");
540
541           #  Line 13
542           printf("\t4) USER LOG MENU    - log rop/loc comments and view lab comments\n");
543
544           #  Line 14
545           printf("\t5) CALL LOAD MENU   - call load tool menu - (under development)\n");
546           #  Line 15
```

```
547         printf(" \n");
548
549         #  Line 16 - 17
550         printf("\tq) quit\n\n");
551
552         #  Line 18
553         printf(" \n");
554
555         #  Line 19
556         printf(" \n");
557
558         #  Line 20
559         printf(" \n");
560
561         #  Line 21 - 22
562         printf("  %s%s\n\n", dash38, dash38);
563
564         #  Line 23
565         printf("Choose an action: ");
566
567         ans = "";
568         getline ans
569         printf("\n");
570
571         if (ans == "")
572             continue;
573
574         if (ans ~ "[Qq]")
575             return;
576
577         if ("12345" !~ ans)
578         {
579          printf("\n□Invalid action: [%s]\n", ans);
580             continue;
581         }
582
583         if (ans ~ "1")
584         {
585             idmenu();
```

```
            continue;
        }

        if (ans ~ "2")
        {
            chgusr();
            continue;
        }

        if (ans ~ "3")
        {
            specopt();
            continue;
        }

        if (ans ~ "4")
        {
            usrlogmenu();
            continue;
        }

        if (ans ~ "5")
        {
            calloadmenu();
        }

    }
}

function imenu()
{
    next_state = "continue";
    while (1)
    {
        if (next_state == "exit")
            return;

        haveids = 0;
        topdisp();
```

```
625
626         #
627         #  Lines 8-9  MENU NAME
628         #
629
630         printf("                                    id MENU\n\n");
631
632         #  Line 10
633         printf("\t1) BEGIN  ID       - logs id      start time to the rop\n");
634
635         #  Line 11
636         printf("\t2) FINISH ID       - logs id      end   time to the rop\n");
637
638         #  Line 12
639         printf("\t3) BEGIN  ID LIST  - logs id list start time to the rop\n");
640
641         #  Line 13
642         printf("\t4) FINISH ID LIST  - logs id list end   time to the rop\n");
643
644         #  Line 14
645         printf(" \n");
646
647         #  Line 15
648         printf("\tq) Return to MAIN MENU\n\n");
649
650         #  Line 16
651         printf(" \n");
652
653         #  Line 17
654         printf(" \n");
655
656         #  Line 18
657         printf(" \n");
658
659         #  Line 19
660         printf(" \n");
661
662         #  Line 20
663         printf(" \n");
```

```
664
665         #  Line 21 - 22
666         printf("  %s%s\n\n", dash38, dash38);
667
668         #  Line 23
669         printf("Choose an action: ");
670
671         ans = "";
672         getline ans
673         printf("\n");
674
675         if (ans == "")
676             continue;
677
678         if (ans ~ "[Qq]")
679             return;
680
681         if ("1234" !~ ans)
682         {
683          printf("\n□Invalid action: [%s]\n", ans);
684             continue;
685         }
686
687         #
688         #  Generate random temporary names - insure unique file names
689         #  for those remote pet users.
690         #
691
692         tempnames();
693
694         if (ans ~ "1")
695         {
696             begid();
697             continue;
698         }
699
700         if (ans ~ "2")
701         {
702             finid();
```

```
703             continue;
704         }
705
706         if (ans ~ "3")
707         {
708             operation = "i";
709             chkidlst();
710
711             continue;
712         }
713
714         if (ans ~ "4")
715         {
716             operation = "u";
717             chkidlst();
718
719             continue;
720         }
721     }
722 }
723
724 function usrlogmenu()
725 {
726     next_state = "continue";
727     while (1)
728     {
729         if (next_state == "exit")
730             return;
731
732         topdisp();
733
734         #
735         #  Lines 8-9  MENU NAME
736         #
737
738         printf("                                    LOG MENU\n\n");
739
740         #  Line 10
741         printf("\t1) LOG COMMENTS       - logs comments to the rop\n");
```

```
742
743             #  Line 11
744             printf("\t2) LOG  LAB PROBLEMS  - logs lab problems found\n");
745
746             #  Line 12
747             printf("\t3) VIEW LAB PROBLEMS  - displays lab problems found\n");
748
749             #  Line 13
750             printf(" \n");
751
752             #  Line 14
753             printf("\tq) Return to MAIN MENU\n\n");
754
755             #  Line 15
756             printf(" \n");
757
758             #  Line 16
759             printf(" \n");
760
761             #  Line 17
762             printf(" \n");
763
764             #  Line 18
765             printf(" \n");
766
767             #  Line 19
768             printf(" \n");
769
770             #  Line 20
771             printf(" \n");
772
773             #  Line 21 - 22
774             printf("  %s%s\n\n", dash38, dash38);
775
776             #  Line 23
777             printf("Choose an action: ");
778
779             ans = "";
780             getline ans
```

```
781             printf("\n");
782
783             if (ans == "")
784                 continue;
785
786             if (ans ~ "[Qq]")
787                 return;
788
789             if ("123" !~ ans)
790             {
791              printf("\n□Invalid action: [%s]\n", ans);
792                 continue;
793             }
794
795             if (ans ~ "1")
796             {
797                 #  Send user comments to rop
798
799                 comment();
800                 continue;
801             }
802
803             if (ans ~ "2")
804             {
805                 #  Log user lab problems for others to look at
806
807                 logprob();
808                 continue;
809             }
810
811             if (ans ~ "3")
812             {
813                 #  View user lab problems
814
815                 viewprob();
816             }
817
818         }
819     }
```

```
820
821  function calloadmenu()
822  {
823       print "\n\tSORRY, this functionallity is still being developed\n";
824       printf("\nHit Enter to return to the MAIN MENU ... ");
825       getline junk
826       return;
827
828       while (1)
829       {
830           topdisp();
831
832           #
833           #  Lines 8-9  MENU NAME
834           #
835
836           #  Line 10
837           printf("                                  CALL LOAD MENU\n");
838
839           #  Line 11
840           printf("\t1) STOP ACS      - stops call load scripts run on ACS\n");
841
842           #  Line 12
843           printf("\t2) RESTORE ACS   - restore scripts run on ACS\n");
844
845           #  Line 13
846           printf("\t3) STOP 2stpe    - brings down the 2stpe process\n");
847
848           #  Line 14
849           printf(" \n");
850
851           #  Line 15
852           printf("\tq) Return to MAIN MENU\n\n");
853
854           #  Line 16
855           printf(" \n");
856
857           #  Line 17
858           printf(" \n");
```

```
859
860         #  Line 18
861         printf(" \n");
862
863         #  Line 19
864         printf(" \n");
865
866         #  Line 20
867         printf(" \n");
868
869         #  Line 21 - 22
870         printf("  %s%s\n\n", dash38, dash38);
871
872         #  Line 23
873         printf("Choose an action: ");
874
875         ans = "";
876         getline ans
877         printf("\n");
878
879         if (ans == "")
880             continue;
881
882         if (ans ~ "[Qq]")
883             return;
884
885         if ("123" !~ ans)
886         {
887          printf("\n□Invalid action: [%s]\n", ans);
888             continue;
889         }
890
891         if (ans ~ "1")
892         {
893             stopACS();
894             continue;
895         }
896
897         if (ans ~ "2")
```

```
898           {
899               restoreACS();
900               continue;
901           }
902
903           if (ans ~ "3")
904           {
905               stop2stpe();
906           }
907       }
908   }
909
910   function stopACS()
911   {
912       ACSpetKey = "/home/acsteam7/bin/acs.key/pet.stop.scripts.key"
913
914       cmd = "ntwrite -t acs68 <" ACSpetKey " 2>&1 > /dev/null; echo $?";
915       cmd | getline exit_code;
916       close(cmd);
917
918       if (exit_code != "0")
919       {
920            printf("\nError: Unable to write to ACS68 ntdeamon.\n");
921       }
922
923
924       cmd = "ntwrite -t acs69 <" ACSpetKey " 2>&1 > /dev/null; echo $?";
925       cmd | getline exit_code;
926       close(cmd);
927
928       if (exit_code != "0")
929       {
930            printf("\nError: Unable to write to ACS69 ntdeamon.\n");
931       }
932   }
933
934   function restoreACS()
935   {
936       ACSpetKey = "/home/acsteam7/bin/acs.key/pet.restore.scripts.key"
```

```
937
938       cmd = "ntwrite -t acs68 <" ACSpetKey " 2>&1 > /dev/null; echo $?";
939       cmd | getline exit_code;
940       close(cmd);
941
942       if (exit_code != "0")
943       {
944            printf("\nError: Unable to write to ACS68 ntdeamon.\n");
945       }
946
947
948       cmd = "ntwrite -t acs69 <" ACSpetKey " 2>&1 > /dev/null; echo $?";
949       cmd | getline exit_code;
950       close(cmd);
951
952       if (exit_code != "0")
953       {
954            printf("\nError: Unable to write to ACS69 ntdeamon.\n");
955       }
956  }
957
958  function stop2stpe()
959  {
960       cmd = "ntwrite -t 2stpe 'exit' 2>&1 > /dev/null; echo $?";
961       cmd | getline exit_code;
962       close(cmd);
963
964       if (exit_code != "0")
965       {
966            printf("\nError: Unable to write to 2stpe ntdeamon.\n");
967       }
968
969       cmd = "ntwrite -t 2stpe 'n' 2>&1 > /dev/null; echo $?";
970       cmd | getline exit_code;
971       close(cmd);
972
973       if (exit_code != "0")
974       {
975            printf("\nError: Unable to write to 2stpe ntdeamon.\n");
```

```
976       }
977
978   }
979
980   function disptinfo()
981   {
982       #  Line 1
983       printf("              Welcome To The (P)ackage (E)nvironment (T)est Tool\n");
984
985       #  Line 2 - 3
986       printf("\n                        Version %s - %s\n", Version, Delta);
987
988       #  Line 4
989       printf("\n");
990
991       #  Line 5 - 6
992       printf("                          *** LATEST UPDATES ***\n\n");
993
994       #  Line 7
995       printf("    ***************  The tstfb tool on ihlpe has been fixed to look at the pet\n");
996
997       #  Line 8
998       printf(" a. *** WARNING ***  database.  Soon, an ERROR will appear denying user\n");
999
000       #  Line 9
001       printf("    ***************  input for ATP ids if the tester did not use the pet tool\n");
002
003       #  Line 10 - 11
004       printf("                     for their package testing.\n\n");
005
006       #  Line 12 - 13
007       printf(" b. A new id ADMIN MENU is available for users to enter id lists\n\n");
008
009       #  Line 13 - 14
010       printf(" c. A LOG USER LOG MENU is available to view/log lab problems.\n\n");
011
012       #  Line 15 - 16
013       printf(" d. The CALL LOAD MENU is still being developed to turn on/off ACS and TARTS\n\n");
014
```

```
015       #  Line 17
016       printf("\n");
017
018       #  Line 18
019       printf("\n");
020
021       #  Line 19
022       printf("\n");
023
024       #  Line 20
025       printf("\n");
026
027       #  Line 21 - 22
028       printf(" ....................................");
029       printf("..................................... \n\n");
030
031       #  Line 23
032       printf("Hit Enter to CONTINUE ... ");
033       getline junk
034  }
035
036  function specopt()
037  {
038       printf("\n");
039       printf("The PET tool will attempt to read the following EXPORTED variables:\n\n");
040       printf("  a.  PHONE   = IXxxxxx | xxxxx | nnn-xxx-xxxx\n\n");
041
042       printf("  b.  DONTASK = y | yes\n\n");
043
044       printf("  c.  EDITOR  = vi | emacs | ed\n\n");
045
046       printf("  d.  PAGER   = more | less | pg | cat\n\n");
047       printf("\n");
048       printf("\nHit Enter to return to the MAIN MENU ... ");
049       getline junk
050  }
051
052  function thankyou()
053  {
```

```
054       printf("\n\n\n");
055       printf("            Thanks For Using The (P)ackage (E)nvironment (T)est Tool\n");
056       printf("\n                          Version %s - %s\n", Version, Delta);
057
058       printf("\n\n");
059       printf(" ....................................");
060       printf("..................................... \n");
061       printf("\n     Send any feedback about this tool to:  %s\n", Provider);
062       printf("\n");
063       printf(" ....................................");
064       printf("..................................... \n");
065       printf("\n\n");
066   }
067
068   function initopen()
069   {
070       #  Intitialize the opnid string
071
072       #
073       #  Check to see if we have any database occurances where we have
074       #  id entries which do not have their end times updated by the user
075       #
076
077       record = "";
078
079       cmd = QUERY Didb " from " idb " where package leq " package;
080       cmd = cmd   " and tindex eq " tindex;
081       cmd = cmd   " and labid leq " labid;
082          cmd = cmd   " and iedate leq " nodate " and ietime leq " notime ;
083
084       idcnt  = 0;
085       dup_rec = 0;
086
087       opnid  = "";
088       while (cmd | getline record)
089       {
090           split(record, a, "|");
091
092           #  If we have entered this function because we just started up
```

```
093           #  there is no need to check for duplicate ids.  This check is
094           #  only necessary for inserting new ids.
095
096           if (id != "")
097               if (id == a[2])
098                   dup_rec = 1;
099
100           if (opnid != "")
101               opnid = opnid " " a[2];
102           else
103               opnid = a[2];
104           idcnt++;
105       }
106       close(cmd);
107
108       #  Again, if we entered this function just because we started pet
109       #  we haven't selected an id yet.
110
111       if (id == "")
112           return;
113
114       if (idcnt > 0)
115           opnid = opnid " " id;
116       else
117           opnid = id;
118       idcnt++;
119
120   }
121
122   function valtn()
123   {
124       tnstat = "ok";
125
126       #
127       #  Strip off IX, IHC, IH or NW prefixes from the tn
128       #
129
130       tnlen = length(tn);
131
```

```
132         for (j=1; j <= tnlen; j++)
133         {
134             byte = substr(tn, j, 1);
135 
136             if (byte ~ "[0-9]")
137             {
138                 ntn = substr(tn, j);
139                 tn = ntn;
140                 break;
141             }
142         }
143 
144         #
145         #  Strip off any dashes
146         #
147 
148         tnlen = length(tn);
149         ntn   = "";
150 
151         for (j=1; j <= tnlen; j++)
152         {
153             byte = substr(tn, j, 1);
154 
155             if (byte != "-")
156                 ntn = ntn byte;
157         }
158         tn = ntn;
159         tnlen = length(tn);
160 
161         if (tnlen == 10)
162         {
163             ntn      = substr(tn, 4);
164             tn       = ntn;
165             tnlen    = 7;
166         }
167 
168 
169         if (tn !~ "[0-9]" ) tnstat = "ng";
170 
```

```
171       if (tnlen == 5)
172       {
173           digit = substr(tn, 1, 1);
174           ext = substr(tn, 2, 4);
175
176           exch = "";
177           if (digit == "3") exch = "713";
178           if (digit == "4") exch = "224";
179           if (digit == "9") exch = "979";
180
181           if (exch == "") tnstat = "ng";
182
183       }
184       else
185       {
186           if (tnlen == 7)
187           {
188               exch = substr(tn, 1, 3);
189            ext  = substr(tn, 4, 4);
190
191               if ((exch != "713") && (exch != "224") && (exch != "979"))
192                   tnstat = "ng";
193           }
194           else
195           {
196               if (tnlen == 12)
197               {
198               }
199               else
200                   tnstat = "ng";
201           }
202       }
203
204       if (tnstat == "ng")
205           tn = "";
206
207       return;
208   }
209
```

```
function getpost()
{

     area = "630";
     pentry = "";

     cmd = "pq -o \"%24pn|%10org|%6oldloc|%6room|%3area %3exch %4ext|%uema\"";
     cmd = cmd " area=" area "/exch=" exch "/ext=" ext " 2>/dev/null";
     cmd | getline pentry;
     close(cmd);

     if (pentry != "")
     {
         split(pentry, a, "|");
         tname  = a[1];
         torg   = a[2];
         tloc   = a[3];
         troom  = a[4];
         ttel   = a[5];
         tmail  = a[6];

         split(tname, b, " ");
         tmpname   = b[1] " " b[2]

         split(tmpname, b, ",");

         shortname = b[2] " " b[1]
         shortpnum = "x" substr(exch, 3,1) ext
     }
     else
     {
         tn       = ""   ;
         tname    = ""   ;
         torg     = ""   ;
         tloc     = ""   ;
         troom    = ""   ;
         ttel     = ""   ;
         tmail    = ""   ;
         addrstat = "ng" ;
```

```
249
250       }
251
252       if (tname != "")
253       {
254           if (dontask == "false")
255           {
256               printf("POST found the following entry:\n\n");
257               tstchk();
258           }
259           else
260           {
261               #
262               #  Expert users know what they are doing
263               #
264
265               addrstat = "ok";
266               sel_tester();
267           }
268       }
269
270       return;
271   }
272
273   function tstchk()
274   {
275       #  If there is no request for a name change and the user requested
276       #  not to be asked if their name is correct, just return
277
278       if ((namechg == "false") && (dontask == "true"))
279       {
280            addrstat = "ok";
281               sel_tester();
282
283            return;
284       }
285
286       printf("%s %s %s %s %s %s\n",
287                 tname, torg, tloc, troom, ttel, tmail);
```

```
288
289         addrstat = "ng";
290         while (1)
291         {
292             printf("\n");
293             printf("> Is the above entry you [n or y (DEFAULT)]? ");
294             ans = "";
295             getline ans;
296
297             if (ans == "")
298                 ans = "y";
299
300             printf("\n");
301
302             if ("yYnN" !~ ans)
303                 continue;
304
305             if ((ans == "y") || (ans == "Y"))
306             {
307                 addrstat = "ok";
308                 namechg  = "false";
309
310                 break;
311             }
312
313             if ((ans == "n") || (ans == "N"))
314             {
315                    tindex  = 0      ;
316                    tname   = ""     ;
317                 torg    = ""     ;
318                    tloc    = ""     ;
319                    troom   = ""     ;
320                    ttel    = ""     ;
321                    tmail   = ""     ;
322                    namechg = "true" ;
323
324                 next_state = "gomain";
325              addrstat   = "ng";
326                 break;
```

```
        }

        if ((ans == "q") || (ans == "Q"))
        {
            next_state = "gomain";
            namechg    = "false";
            break;
        }
    }

    if (addrstat == "ok")
    {
        sel_tester();
    }

    return;
}

function idchk()
{

    printf("%s\n", usr_msg);

    while (1)
    {
        printf("\n");
        printf("> Is the above id NUMBER correct [n on y (DEFAULT)]? ")
        ans = "";
        getline ans;

        if (ans == "")
            break;

        printf("\n");

        if ("yYnN" !~ ans)
            continue;

        if ((ans == "y") || (ans == "Y"))
```

```
366             {
367                 break;
368             }
369
370             if ((ans == "n") || (ans == "N") || (ans == "q") || (ans == "Q"))
371             {
372                    id = "";
373
374                 if ((ans == "q") || (ans == "Q"))
375                     next_state = "gomain";
376
377                 break;
378             }
379         }
380         return;
381     }
382
383     function getphone()
384     {
385         while (1)
386         {
387             printf("\n");
388             printf("> Enter your work phone number (i.e. 91234): ");
389
390             ntn = "";
391             getline ntn
392             printf("\n");
393
394             if (ntn == "")
395                 continue;
396
397             if (ntn == tn)
398             {
399                 printf("\n");
400                 printf("Previous work phone number ane new work phone are IDENTICAL\n");
401                 continue;
402             }
403
404             #
```

```
405             #  Did the user just want to return to the main window
406             #
407
408             if ((ntn == "q") || (ntn == "Q"))
409             {
410                 next_state = "gomain";
411                 return;
412             }
413
414             #
415             #  Set tn the the new tn just read, but save the original
416             #  value in case valtn() flags it as a bad tn.
417             #
418
419             origtn = tn;
420             tn     = ntn;
421
422             valtn();
423
424             if (next_state != "continue")
425             {
426                 tn = origtn;
427                 return;
428             }
429
430             if (tnstat == "ng")
431             {
432                 printf("\nError:  Invalid work phone number [%s]□\n", tn);
433
434                 #
435                 #  Set the tn back to the original value
436                 #
437
438                 tn = origtn;
439             }
440             else
441             {
442                 getpost();
443
```

```
444                 if (next_state != "continue")
445                 {
446                     tn = origtn;
447                     return;
448                 }
449 
450                 if (addrstat == "ok")
451                     break;
452                 else
453                 {
454                     tn  = origtn;
455                     ans = "";
456                 }
457             }
458 
459         }   #  End of while loop
460 
461         return;
462     }
463 
464     function getpkgno()
465     {
466         package = "";
467         bwmnofn = "";
468 
469         #
470         #  Are we doing a remote pet entry and the bwmno file exists ?
471         #
472 
473         cmd = "ls /home/surat/rje/uucpdir/bwmno 2>/dev/null";
474         cmd | getline bwmnofn;
475         close(cmd);
476 
477         if (bwmnofn != "")
478         {
479             #
480             #  Move the current files to the previous directory and install
481             #  new bwmno current file.
482             #
```

```
483
484         cmd = "mv /home/surat/reglib/current/* /home/surat/reglib/previous/.";
485         system(cmd);
486         close(cmd);
487
488         cmd = "cp /home/surat/rje/uucpdir/bwmno ";
489         cmd = cmd "/home/surat/reglib/current/bwmno";
490         system(cmd);
491         close(cmd);
492
493         cmd = "rm -f /home/surat/rje/uucpdir/bwmno";
494         system(cmd);
495         close(cmd);
496
497         cmd = "cp /home/surat/rje/uucpdir/packname ";
498         cmd = cmd "/home/surat/reglib/current/packname";
499         system(cmd);
500         close(cmd);
501
502         cmd = "rm -f /home/surat/rje/uucpdir/packname";
503         system(cmd);
504         close(cmd);
505
506         cmd = "cp /home/surat/rje/uucpdir/pcc ";
507         cmd = cmd "/home/surat/reglib/current/pcc";
508         system(cmd);
509         close(cmd);
510
511         cmd = "rm -f /home/surat/rje/uucpdir/pcc";
512         system(cmd);
513         close(cmd);
514
515         cmd = "cp /home/surat/rje/uucpdir/pagerlist ";
516         cmd = cmd "/home/surat/reglib/current/pagerlist";
517         system(cmd);
518         close(cmd);
519
520         cmd = "rm -f /home/surat/rje/uucpdir/pagerlist";
521         system(cmd);
```

```
522         close(cmd);
523
524         cmd = "cp /home/surat/rje/uucpdir/maillist ";
525         cmd = cmd "/home/surat/reglib/current/maillist";
526         system(cmd);
527         close(cmd);
528
529         cmd = "rm -f /home/surat/rje/uucpdir/maillist";
530         system(cmd);
531         close(cmd);
532
533     }
534
535     cmd = "tail -1 " curpkg " 2>/dev/null";
536
537     record = "";
538     cmd | getline record;
539     close(cmd);
540
541     if ((suvlab == "y") && (record == ""))
542     {
543         printf("□\n");
544         printf("FATAL ERROR:  Can't get current package under test (ec:1a)\n");
545         printf("              Contact Tool Provider: %s\n", Provider);
546         printf("□\n");
547
548         next_state = "exit";
549         return;
550     }
551
552     if (record == "")
553     {
554         promptpack();
555
556         if (package == "")
557         {
558             next_state = "exit";
559         }
560     }
```

```
561       else
562           package = record;
563   }
564
565   function getidno()
566   {
567       while (1)
568       {
569           printf("\n");
570           printf("> Enter your id NUMBER (i.e. 1234567): ");
571
572           id = "";
573           getline id;
574           printf("\n");
575
576           if (id != "")
577           {
578               #  Did the user just want to return to the main window
579
580               if ((id == "q") || (id == "Q"))
581               {
582                   next_state = "gomain";
583                   id = "";
584
585                   return;
586               }
587
588               if ((id ~ /^[1-9][0-9][0-9][0-9][0-9][0-9]$/)      ||
589                   (id ~ /^[1-9][0-9][0-9][0-9][0-9][0-9][0-9]$/))
590               {
591                   break;
592               }
593               printf("\nError:  Invalid id NUMBER [%s]□\n", id);
594           }
595       }
596   }
597
598   function promptpack()
599   {
```

```
600         package = "";
601         while (1)
602         {
603             printf("\n");
604             printf("> Enter a PACKAGE NUMBER (i.e. 3z07): ");
605 
606             record = "";
607             getline record;
608             printf("\n");
609 
610             if ((record ~ /^3[azi][0-9][0-9]$/)       ||
611                 (record ~ /^3[azi][0-9][a-z]$/)       ||
612                 (record ~ /^3[azi][a-z][0-9]$/)       ||
613                 (record ~ /^3[azi][a-z][a-z]$/)       ||
614                 (record ~ /^[0-9][0-9]0[1-9]$/)       ||
615                 (record ~ /^[0-9][0-9][1-9][0-9]$/)   ||
616                 (record ~ /^A[0-9]]0[1-9]$/)          ||
617                 (record ~ /^A[0-9]][1-9][0-9]$/)      ||
618                 (record ~ /^Z[0-9]]0[1-9]$/)          ||
619                 (record ~ /^Z[0-9]][1-9][0-9]$/)      ||
620                 (record ~ /^I[0-9]]0[1-9]$/)          ||
621                 (record ~ /^I[0-9]][1-9][0-9]$/))
622             {
623 
624                 break;
625             }
626 
627             if ((record == "q") || (record == "Q"))
628                 return;
629 
630             printf("\nError:  Invalid PACKAGE NUMBER [%s]□\n", record);
631         }
632 
633         cmd = "date +%y";
634         cmd | getline year;
635         close(cmd);
636 
637         package = "bwm" year "-" record;
638     }
```

```
function getusrin()
{
     #
     #  This function will collect user information
     #

     next_state = "continue";

     if (tname == "")
     {
         getphone();

         if (next_state != "continue")
             return;
     }
     else
     {
         if (dontask == "true")
         {
             sel_tester();
             return;
         }

         printf("%s:\n\n", usr_msg);
         tstchk();

         if (next_state != "continue")
             return;

         if (tname == "")
         {
             getphone();

             if (next_state != "continue")
                 return;
         }
     }
```

```
678       #
679       #  Make certain you have a tester index
680       #
681
682       if (tindex == 0)
683       {
684           sel_tester();
685       }
686   }
687
688   function sel_tester()
689   {
690       record = "";
691
692       cmd = QUERY Dtdb " from " tdb " where ttel leq \"" ttel "\"";
693       cmd | getline record;
694       close(cmd);
695
696       if (record != "")
697       {
698           split(record, a, "|");
699           tindex = a[2];
700           torg   = a[3];
701           tloc   = a[4];
702           troom  = a[5];
703           ttel   = a[6];
704           tmail  = a[7];
705       }
706       else
707       {
708           krecord = "";
709
710           cmd = QUERY Dtidx " from " tidx " where tindex ge 0";
711           cmd | getline krecord;
712           close(cmd);
713
714           if (krecord == "")
715           {
716               printf("□\n");
```

```
717             printf("FATAL DATABASE ERROR:  Index db returned no index (ec:3)\n");
718             printf("□\n");
719             exit(1);
720         }
721         else
722         {
723             tindex = krecord;
724             tindex++;
725 
726             cmd = UPDATE Dtidx " tindex to " tindex " in " tidx ;
727             cmd = cmd " where tindex eq " krecord;
728 
729             if ((system(cmd)) != 0)
730             {
731                 printf("\n□\n");
732                 printf("FATAL ERROR:  Database failure while updating tstindx.\n");
733                 printf("              Update ABORTED (ec:4).\n\n");
734 
735                 printf("\n□\n");
736 
737                 next_state = "exit";
738                 close(cmd);
739                 return;
740 
741             }
742             close(cmd);
743 
744             print tname "|" tindex "|" torg "|" tloc "|" troom "|" ttel "|" tmail > tfile ;
745             close(tfile);
746 
747             cmd = INSERT Dtdb " in " tdb " from " tfile;
748 
749             if ((system(cmd)) != 0)
750             {
751                 printf("\n□\n");
752                 printf("FATAL ERROR:  Database failure while inserting testerdb.\n");
753                 printf("              Insert ABORTED (ec:5).\n\n");
754 
755                 printf("\n□\n");
```

```
                next_state = "exit";

            }
            close(cmd);
            cmd = "rm -f " tfile;

            system(cmd);
            close(cmd);
        }
    }

}

function ins_id()
{
    #
    #  This function will INSERT the id entries for a user.
    #

    initopen();

    if (dup_rec == 1)
    {
        printf("\n□\n");
        printf("ERROR:  Duplicate record detected while inserting iddb.\n");
        printf("              Insert ABORTED (ec:6).\n\n");

        printf("\n□\n");

        next_state = "gomain";
        return;
    }

    datecmd | getline isdate;
    close(datecmd);

    timecmd | getline istime;
    close(timecmd);
```

```
795
796       iedate = nodate;
797       ietime = notime;
798
799          print package "|" id "|" tindex "|" labid "|" isdate "|" istime "|" iedate "|" ietime >
800   tfile ;
801       close(tfile);
802
803       cmd = INSERT Didb " in " idb " from " tfile;
804
805       if ((system(cmd)) != 0)
806       {
807           printf("\n□\n");
808           printf("FATAL ERROR:  Database failure while inserting iddb.\n");
809           printf("              Insert ABORTED (ec:7).\n\n");
810
811           printf("\n□\n");
812
813           close(cmd);
814
815           next_state = "exit";
816           return;
817
818       }
819       close(cmd);
820
821       cmd = "rm -f " tfile;
822       system(cmd);
823       close(cmd);
824   }
825
826   function upd_id()
827   {
828       #
829       #  This function will UPDATE the id entries for a user.
830       #
831
832       #
833       #  Check to see if the user did a BEGIN id for this package
```

```
834         #
835
836         record = "";
837
838         cmd = QUERY Didb " from " idb " where package leq " package ;
839         cmd = cmd   " and tindex eq " tindex ;
840         cmd = cmd   " and id eq " id ;
841            cmd = cmd   " and iedate leq " nodate " and ietime leq " notime ;
842
843         rec_cnt = 0;
844         while (cmd | getline record)
845         {
846             rec_cnt++;
847         }
848         close(cmd);
849
850         if (rec_cnt == 0)
851         {
852             printf("\n□\n");
853             printf("ERROR:  Package Tester: [%s]\n", tname);
854             printf("        For Package:    [%s]\n", package);
855             printf("        Did not execute an BEGIN id command.\n\n");
856
857                printf("Please execute the BEGIN id command first.\n");
858             printf("\n□\n");
859
860             id        = "";
861             next_state = "gomain";
862             return;
863         }
864
865         #  Update the id count and open id array
866
867         idcnt = split(opnid, a, " ");
868         cnt    = idcnt;
869         tmp_opn = "";
870         for (i=1; i <= cnt; i++)
871             if (a[i] == id)
872                 idcnt--;
```

```
873         else
874             if (tmp_opn == "")
875                     tmp_opn = a[i];
876             else
877                     tmp_opn = tmp_opn " " a[i];
878
879     opnid = tmp_opn;
880
881     #
882     #  Update the database id entry
883     #
884
885     datecmd | getline iedate;
886     close(datecmd);
887
888     cmd = UPDATE Didb " iedate to " iedate
889        cmd = cmd    " in " idb ;
890     cmd = cmd    " where tindex  eq " tindex ;
891     cmd = cmd    " and  package leq " package;
892     cmd = cmd    " and  id eq " id;
893
894     if ((system(cmd)) != 0)
895     {
896         printf("\n□\n");
897         printf("FATAL ERROR:  Database failure while updating iddb.\n");
898         printf("              Update ABORTED (ec:9).\n\n");
899
900         printf("\n□\n");
901
902         close(cmd);
903
904         next_state = "exit";
905         return;
906
907     }
908     close(cmd);
909
910     timecmd | getline ietime;
911     close(timecmd);
```

```
912
913       cmd = UPDATE Didb " ietime to " ietime
914          cmd = cmd    " in " idb ;
915       cmd = cmd    " where tindex eq " tindex ;
916       cmd = cmd    " and  package leq " package;
917       cmd = cmd    " and  id eq " id;
918
919       if ((system(cmd)) != 0)
920       {
921           printf("\n□\n");
922           printf("FATAL ERROR:  Split transaction occurred in iddb.\n");
923           printf("              Update ABORTED (ec:10).\n\n");
924
925           printf("\n□\n");
926
927           close(cmd);
928
929           next_state = "exit";
930           return;
931       }
932       close(cmd);
933
934       if (nfslab == "n")
935       {
936           operation = "u";
937           updpetdb();
938       }
939   }
940
941
942   function chkidlst()
943   {
944       #
945       #  This function will get the id list from the user and update the
946       #  pet database.
947
948       while (1)
949       {
950           #
```

```
951         #  Get the users id file name
952         #
953
954         getfn();
955
956         if (userfn == "")
957             return;
958         else
959         {
960             #
961             #  Is the file real?
962             #
963
964             cmd = "ls " userfn " 2>/dev/null";
965             cmd | getline results;
966             close(cmd);
967
968             if (results == "")
969             {
970                 printf("\nError:  id File [%s] does not exist!\n", userfn);
971          }
972          else break;
973         }
974       }
975
976       #
977       #  Validate the ids contained in the file
978       #
979
980       validlst();
981
982       #
983       #  If we have a string of ids, then process them.
984       #
985
986       for (loop=1; loop <= usridcnt; loop++)
987       {
988           haveid = 1;
989           id     = usrid[loop];
```

```
990
991             #
992             #  Insert or Update the users id into the database
993             #
994
995             if (operation == "i")
996                 begid();
997             else
998                 finid();
999
1000        }
1001        haveid = 0;
1002    }
1003
1004    function validlst()
1005    {
1006        #
1007        #  This function will validate the ids in the user file
1008        #  Users can have only one id per line.
1009        #
1010
1011        usridcnt = 0 ;
1012        while ((getline aid < userfn) > 0)
1013        {
1014            if ((aid ~ /^[1-9][0-9][0-9][0-9][0-9][0-9]$/)      ||
1015                (aid ~ /^[1-9][0-9][0-9][0-9][0-9][0-9][0-9]$/))
1016            {
1017                usridcnt++;
1018                usrid[usridcnt] = aid;
1019            }
1020            else
1021                printf("\nError:  Invalid id NUMBER [%s]□\n", aid);
1022
1023        }
1024        close(userfn);
1025    }
1026
1027    function getfn()
1028    {
```

```
029     if (operation == "i")
030     {
031         msg = "  |                              * * * BEGIN id * * *";
032         msg = msg "                              |\n";
033         msg = msg "  |                                                ";
034         msg = msg "                              |";
035     }
036     else
037     {
038         msg = "  |                               * * * END id * * * ";
039         msg = msg "                              |\n";
040         msg = msg "  |                                                ";
041         msg = msg "                              |";
042     }
043
044     while (1)
045     {
046         printf("\n  %s%s\n", dash38, dash38);
047         printf("%s\n", msg);
048         printf("  | You can now enter a filename containing your list ");
049         printf("of ids.  Your file    |\n");
050
051         printf("  | must have only 1 id per line and have no NULL ");
052         printf("lines.                     |\n");
053
054         printf("  |                                                  ");
055         printf("                          |\n");
056
057         printf("  | The program will expand exported $ variables and");
058         printf(" will expand ~           |\n");
059         printf("  %s%s\n\n", dash38, dash38);
060
061         printf("Enter your the FILENAME containing your ids:\n\n> ");
062
063         yourfn = "";
064         getline yourfn;
065         printf("\n");
066
067         if (yourfn != "")
```

```
068         {
069             #  Did the user just want to return to the main window
070
071             if ((id == "q") || (id == "Q"))
072             {
073                 next_state = "gomain";
074                 userfn = "";
075
076                 return;
077             }
078
079             #
080             #  Resolve any relative path from $HOME or ~ variables
081             #
082
083             items = split(yourfn, a, "/");
084
085             yourpath = "";
086             for (i=1; i <= items; i++)
087             {
088                 if (substr(a[i], 1, 1) == "$")
089                 {
090                     shellvar = substr(a[i], 2);
091                     getdollar();
092
093                     if (yourpath == "")
094                         yourpath = vardef;
095                     else
096                         yourpath = yourpath "/" vardef;
097                 }
098                 else
099                 {
100                     if (substr(a[i], 1, 1) == "~")
101                     {
102                         shellvar = substr(a[i], 2);
103                         gettilde();
104
105                         if (yourpath == "")
106                             yourpath = vardef;
```

```
                        else
                            yourpath = yourpath "/" vardef;
                    }
                    else
                    {
                        if (yourpath == "")
                            yourpath = a[i];
                        else
                            yourpath = yourpath "/" a[i];
                    }
                }
            }
            break;
        }
    }
    userfn = yourpath;
}

function begid()
{
    #
    #  This function will excute a BEGIN id option for the user
    #

    if (haveid == 0)
    {
        usr_msg = "Ready to BEGIN id for";

        getusrin();

        if (next_state != "continue")
            return;

        getidno();

        if (next_state != "continue")
            return;
    }
    else
```

```
146       {
147           printf("\nReady to BEGIN id TESTING for id: %s\n", id);
148       }
149
150       ins_id();
151
152       if (next_state != "continue")
153           return;
154
155       printf("\nLogging BEGIN id to the rop ...");
156
157       ROPMSG = sprintf("+++ BEGIN id +++  %s  %s  id: %s",
158                        shortname, shortpnum, id);
159
160       log3bmsg();
161
162       printf("\n");
163
164       if (nfslab == "n")
165       {
166           #  Were a remote user -- update the official pet database
167
168           operation = "i";
169           updpetdb();
170       }
171   }
172
173   function finid()
174   {
175       #
176       #  This function will excute a FINISH id option for the user
177       #
178
179       if (haveid == 0)
180       {
181           usr_msg = "Ready to FINISH id for: ";
182           getusrin();
183
184           if (next_state != "continue")
```

```
185             return;
186
187         if (id == "")
188         {
189             getidno();
190         }
191         else
192         {
193             usr_msg = "Ready to FINISH id TESTING for id: " id;
194             idchk()
195
196             if (next_state != "continue")
197                 return;
198
199             if (id == "")
200                 getidno();
201         }
202
203         if (next_state != "continue")
204             return;
205     }
206     else
207     {
208         printf("\nReady to FINISH id TESTING for id: %s\n", id);
209     }
210
211     upd_id();
212
213     if (next_state != "continue")
214         return;
215
216     printf("\nLogging FINISH id to the rop ...");
217
218     ROPMSG = sprintf("+++ FINISH id +++  %s  %s  id: %s",
219                      shortname, shortpnum, id);
220
221     log3bmsg();
222
223     printf("\n");
```

```
}

function chgusr()
{
     namechg = "true";

     otn     = tn      ;
     otname  = tname   ;
     otorg   = torg    ;
     otloc   = tloc    ;
     otroom  = troom   ;
     ottel   = ttel    ;
     otindex = tindex  ;

     tn      = "" ;
     tname   = "" ;

     printf("\n  %s%s\n", dash38, dash38);
     msg = "  |                          * * * CHANGE USER * * *   ";
     msg = msg "                        |";
     print msg;
     printf("  %s%s\n", dash38, dash38);


     getusrin();

     if (next_state == "gomain")
     {
         #
         #  User changed their mind
         #

         namechg = "false"  ;
         tn      = otn      ;
         tname   = otname   ;
         torg    = otorg    ;
         tloc    = otloc    ;
         troom   = otroom   ;
         ttel    = ottel    ;
```

```
263         tindex  = otindex  ;
264     }
265     else
266     {
267         id = "";
268         initopen();
269     }
270 }
271
272 function comment()
273 {
274
275     ROPMSG = sprintf("+++ BEGIN COMMENTS +++  %s  %s",
276                      shortname, shortpnum);
277     print ROPMSG > cfile;
278     close(cfile);
279
280     printf("\n");
281     printf("Enter the comments you wish to display to the rop (max 60 characters/line)\n");
282     printf("      A carriage return with no input ends comments\n");
283     printf("............................................................\n");
284
285     gavecomment = 0;
286     while (1)
287     {
288         ans = "";
289         getline ans;
290
291         if (ans == "")
292             break;
293         else
294         {
295             gavecomment = 1;
296             trim = substr(ans, 1, 60);
297          print trim >> cfile;
298          close(cfile);
299         }
300     }
301     if (gavecomment == 0)
```

```
302       {
303           cmd = "rm -f " cfile;
304           system(cmd);
305           close(cmd);
306       }
307       else
308       {
309           ROPMSG = sprintf("+++ END COMMENTS +++  %s  %s",
310                            shortname, shortpnum);
311           print ROPMSG >> cfile;
312           close(cfile);
313
314           log3bcmt();
315       }
316   }
317
318   function log3bcmt()
319   {
320       #
321       #  This function will send a file of comments to the rop
322       #
323
324       if (debug == "ON")
325           cmd = "nohup MSGFILE=" cfile " writerop" ;
326       else
327           cmd = "nohup MSGFILE=" cfile " " home "/regbin/writerop";
328
329       cmd = cmd " > /dev/null 2>&1 &";
330       system(cmd);
331       close(cmd);
332   }
333
334   function log3bmsg()
335   {
336
337       #
338       #  This function will send to the 3B a message for the rop
339       #
340
```

```
341       if (debug == "ON")
342           cmd = "MSG=\"" ROPMSG "\"  writerop";
343       else
344           cmd = "MSG=\"" ROPMSG "\" " home "/regbin/writerop";
345
346       system(cmd);
347       close(cmd);
348   }
349
350   function logprob()
351   {
352       #
353       #  Log a lab problem found by a user
354       #
355
356       cmd = "echo $EDITOR";
357       cmd | getline ed;
358       close(cmd);
359
360       while (1)
361       {
362
363           if (ed == "")
364           {
365               printf("Your EDITOR variable is currently not set\n\n");
366               printf("ENTER the name of the EDITOR you would like to use: " );
367
368               ans = "";
369               getline ans
370               print " ";
371           }
372           else
373           {
374               if (dontask != "true")
375               {
376                   printf("ENTER the name of the EDITOR [default %s] to use: ", ed);
377                   ans = "";
378                   getline ans
379                   print " ";
```

```
            }
            else
                ans = ed;
        }


        if (ans != "")
        {
            n = split(ans, a, "/");
            ed = a[n];

            if (ed ~ "vi" || ed ~ "emacs" || ed ~ "ed")
                break;

         printf("\n□\nSorry, ");
            printf("Don't know about that editor, try vi, emacs, or ed\n");

            ed = "";
        }
    }
    cmd = "( touch " efile " ; chmod 777 " efile ";"
    cmd = cmd rmsuid " " ed " " efile " )";
    system(cmd);
    close(cmd);

    cmd = "date";
    cmd | getline timestamp;
    close(cmd);

    split(ttel, a, " ");
    wrktn = a[2] "-" a[3];

    cmd = "echo \"\n" timestamp " - " tname " - " wrktn "\" >> " lprobdb;
    system(cmd);
    close(cmd);

    cmd = "cat " efile " >> " lprobdb;
    system(cmd);
    close(cmd);
```

```
419
420       cmd = "rm -f " efile;
421       system(cmd);
422       close(cmd);
423   }
424
425   function viewprob()
426   {
427       #
428       #  View the lab problems found
429       #
430
431
432       cmd = "ls " lprobdb " 2>/dev/null";
433       cmd | getline problem;
434       close(cmd);
435
436       if (problem == "")
437       {
438           printf("\n");
439           printf("\t\t-----------------------------------\n");
440           printf("\t\tCURRENT LAB PROBLEMS for %s\n", package);
441           printf("\t\t-----------------------------------\n\n");
442           printf("\t\t               NONE\n");
443       }
444       else
445       {
446           cmd = "echo $PAGER";
447           cmd | getline pg;
448           close(cmd);
449
450           while (1)
451           {
452               if (pg == "")
453               {
454                   printf("Your PAGER variable is currently not set\n\n");
455                   printf("ENTER the name of the PAGER you would like to use: " );
456                   ans = "";
457                   getline ans
```

```
458             print " ";
459         }
460         else
461         {
462             if (dontask != "true")
463             {
464                 printf("ENTER the name of the PAGER [default %s] to use: ", pg);
465                 ans = "";
466                 getline ans
467                 print " ";
468             }
469             else
470                 ans = pg;
471         }
472
473         if (ans != "")
474         {
475             n = split(ans, a, "/")
476             ed = a[n]
477
478             if (pg ~ "more" || pg ~ "less" || pg ~ "cat" || pg ~ "pg")
479                 break;
480
481          printf("\n□\nSorry, ");
482             printf("Don't know about that PAGER, TRY: more, less, cat or pg\n");
483             pg = "";
484         }
485     }
486
487     printf("\n");
488     printf("\t\t-----------------------------------\n");
489     printf("\t\tCURRENT LAB PROBLEMS for %s\n", package);
490     printf("\t\t-----------------------------------\n\n");
491
492     cmd = pg " " lprobdb;
493     system(cmd);
494     close(cmd);
495 }
496
```

```
497       printf("\n");
498       printf("\t\t-----------------------------------\n");
499       printf("\t\tEND OF  LAB PROBLEMS for %s\n", package);
500       printf("\t\t-----------------------------------\n\n");
501
502       printf("Hit Enter to return to the LOG MENU ... ");
503       getline junk
504   }
505
506   function tempnames()
507   {
508
509       #  Build the currstr because %.% is evaluated by SCCS and produces
510       #  undesireable results.
511
512       currstr = "date +%m"   ;
513          currstr = currstr "%d" ;
514          currstr = currstr "%H" ;
515          currstr = currstr "%M" ;
516          currstr = currstr "%S" ;
517
518       secscmd | getline cursecs;
519       close(secscmd);
520
521       multiplier = 100 + cursecs;
522
523       currstr | getline curtime;
524       close(currstr);
525
526          tpfx    = sprintf("%03d.%d", rand() * multiplier, curtime);
527
528          tfile   = sprintf("/usr/tmp/tpet.%d.%s", tindex, tpfx);
529          cfile   = sprintf("/usr/tmp/cpet.%d.%s", tindex, tpfx);
530          efile   = sprintf("/usr/tmp/epet.%d.%s", tindex, tpfx);
531   }
532
533   function updpetdb()
534   {
535       #
```

```
536         #  Update the pet database on the lts machine
537         #
538
539            tranfile   = sprintf("%s/pet_xfer/PET.%s.%d.%d",
540                                home, operation, tindex, curtime);
541
542         record = "";
543         cmd = QUERY Dtdb " from " tdb " where tindex eq " tindex;
544         cmd | getline record;
545         close(cmd);
546
547         if (record != "" )
548         {
549             print record >> tranfile;
550             close(tranfile);
551         }
552         else
553         {
554             print "WARNING:  The pet database was NOT updated!" >> tranfile;
555             print "          tstfb will have no record of your test session\n" >> tranfile;
556             print "          Please contact the pet adminstrator ASAP!\n" >> tranfile;
557             print "PROBLEM:  Can't Query package tester from package test data base" >> tranfile;
558             print "          The query command used was: \n" >> tranfile;
559             print cmd >> tranfile;
560             close(tranfile);
561
562             mailcmd = "mail -s \"updpetdb FAILURE\" < " tranfile;
563             system(mailcmd);
564             close(cmd);
565
566             cmd = "rm -f " tranfile;
567             system(cmd);
568             close(cmd);
569
570             return;
571         }
572
573         record = "";
574         cmd = QUERY Didb " from " idb " where package leq " package ;
```

```
575       cmd = cmd   " and tindex eq " tindex ;
576       cmd = cmd   " and id eq " id ;
577          cmd = cmd   " and iedate leq " iedate " and ietime leq " ietime ;
578
579       cmd | getline record;
580       close(cmd);
581
582       if (record != "" )
583       {
584           print record >> tranfile;
585           close(tranfile);
586       }
587       else
588       {
589           print "WARNING:  The pet database was NOT updated!" >> tranfile;
590           print "          tstfb will have no record of your test session\n" >> tranfile;
591           print "          Please contact the pet adminstrator ASAP!\n" >> tranfile;
592           print "PROBLEM:  Can't Query id testing session from the data base" >> tranfile;
593           print "          The query command used was: \n" >> tranfile;
594           print cmd >> tranfile;
595           close(tranfile);
596
597           mailcmd = "mail -s \"updpetdb FAILURE\" < " tranfile;
598           system(mailcmd);
599           close(mailcmd);
600
601           cmd = "rm -f " tranfile;
602           system(cmd);
603           close(cmd);
604       }
605   }
```