NAME

au - au or assembly unit file

DESCRIPTION

The au file is a compact packet of control information used to accompany any programs which are incorporated into an SCCS Generic. The information collected in an au file represents a pident. The au file name is formed from a concatenation of the pident name and the string, ".au". The au file is broken into five sections of the following names and functions:

#IDENTIFICATION
The IDENTIFICATION section contains identifying information of the pident. See below.

#PROGRAM UNITS
The PROGRAM UNITS section contains the printable (i.e. ascii) files associated with the pident. The entries in this section are listed one per line with an optional title following the file name separated by blanks or tabs. These entries are picked up by the plistp command and printed.

#DATA
The DATA section contains nonprintable files associated with the pident. The entries are listed one per line with an optional title following the file name separated by tabs or blanks. Patterns and libraries are typically listed in this section.

#MAKE
The MAKE section contains information on how the pident's source is made, i.e., compiled, assembled, loaded, archived, etc., into an object module. This section consists of commands that shell can execute.

#COPY
The COPY section has shell commands which move the made object module(s) to the final residing place on the produced generic. Two commands, cpmv and move, have been written for this purpose.

The ordering of the sections within the file is important to certain administrative programs and thus the above sequence is recommended. The #IDENTIFICATION and #PROGRAM UNITS sections must be part of every au file and ordered first and second, respectively, within the file.

The IDENTIFICATION section is made up of subfields with the following names and functions:

NAME	The name of the pident.
DOC	The PR number the pident is associated with.
ISSUE	The issue of the pident.

DATE	The date the pident was last issued.
OWNER	The programmer responsible for the pident (his login id).

To assist the developer in the creation of the au file a prompt procedure exists. To use it, the programmer should perform the following UNIX commands:

```
chdir /pst/gadm/aumake
form au
```

The user will be prompted 8 times for the necessary information. The user should be familiar with the program _form_ before attempting this procedure.

FILES

SEE ALSO

cpmv(I), plistp(I), secprt(I), move(I)

NAME

chldata - Channel Data File

DESCRIPTION

The channel data file, chldata, describes certain information about each channel on the SCCS. The file contains MAXCHL fixed - size records, and it is intended that information be extracted from the file by reading the file into an array of "CHL_B" structures, or by reading only a portion of the file into one such structure. The subroutine idchl(III) will return the information for a single channel.

All information in the file is in ascii; hence all elements are defined to be character arrays. The file is initialized to contain spaces in all elements, except the element "c_end", which is initialized to contain the string 'x\n' (so the file may be printed). It should be noted that all elements start on an even byte boundary; this causes some elements to be longer than they would have to be. In general, data is left - justified in each element, and leading zeroes appear where necessary to insure the required precision. Spaces are used to pad out unneeded characters in the elements c_name, c_act, c_gen, c_issue, and c_issno.

The element "c_act" is a special element; it is reserved for possible use by the recent change programs. It should be ignored by all other programs.

The element "c_issno" is a conversion of the element "c_issue" which allows the latter to always be represented as a number. The conversion is performed by multiplying the numeric portion of c_issue by decimal 10 and adding a number 1 through 9 to represent the possible letter issues "a" through "i". If no letter issue exists, nothing is added. For example, the following c_issue elements convert to the following c_issno elements:

c_issue	c_issno
01	010
05	050
10	100
10a	101
14c	143

NAME

gen_rng - Generic Range File

DESCRIPTION

The generic range file, gen rng, resides in an appropriate /type?? directory for each switching machine (SPCS) for which any one of the following SCCS features are supported:

RC:BUILD
Scheduled Common Analysis
Demand Common Analysis
3B Common Processor Features

The generic range file contains a number of entries that identify one or more groups of routines to be executed for each of the above features. Which group of routines should be executed is determined by such things as the feature to be performed, the feature function to be performed, and the series of SPCS generics and issues that is pertinent to the office for which the requested task is being performed.

Each generic range file entry has a fixed size and has the structure **GEN_RNG**, as defined in the header file, gen rng.h. All information in the entry is in ASCII; hence all elements are defined as character strings. Each entry must be initialized to contain blanks in all elements or unused portions of elements that do not contain data. Data in an entry is left-justified in each element.

The elements **gr_fgen** and **gr_tgen** specify the "from" and "to" generic ID's that are to be used for range checking. **Gr_fgen** specifies the lower bound and must always contain either an entire generic ID or the first few digits of the generic ID that identify the generic base. **Gr_tgen** specifies the upper bound and may contain an entire generic ID or the first few digits of the generic ID that identify the generic base. **Gr_tgen** may also contain a '-' to indicate that all generic ID's that are greater than or equal to **gr_fgen** are to be accepted. Note that if the value specified for **gr_fgen** contains only a generic base, then the value for **gr_tgen** must also contain only a generic base or a '-'.

The elements **gr_fiss** and **gr_tiss** specify the "from" and "to" abstract issue numbers that are to be used for range checking. These elements normally contain the value '-' except when it becomes necessary to perform range checking on an issue basis rather than on a generic basis. In such cases, the element **gr_fiss** identifies the "from" abstract issue number that specifies the lower bound for the range checking and the element **gr_tiss** identifies the "to" abstract issue number that specifies the upper bound. A '-' entry for **gr_tiss** means that all abstract issue numbers for the indicated generic ID that are "greater than or equal to" **gr_fiss** are to be accepted.

If it becomes necessary to perform range checking on an issue basis for a certain feature and function, the following steps must be followed:

1. New entries must be inserted into the generic range file for the affected feature, function, and generic ID. These new entries must specify the appropriate range of abstract issue numbers that are served by the routines specified in the generic range file entry.

2. Be certain to remove old entries, that pertain to the affected feature and function, from the generic range file.

The following is a listing of the _gen_rng.h_ header file.

```
/*
    This header file defines the structure for the "gen_rng" file
    presently used by RC:BUILD and COMMON ANALYSIS distributor
    routines to determine which routines must be executed to per-
    form the desired functions.  The programs to be executed are
    determined by the office generic and issue.
*/


/*
    Define the name of the generic range file.
*/

#define GEN_RNG_FIL "gen_rng"


/*
    Define supported features.
*/

#define GR_RCBLD     "rcb"    /* RC:BUILD */
#define GR_SCA       "sca"    /* Scheduled COMMON ANALYSIS Routines */
#define GR_DCA       "dca"    /* Demand COMMON ANALYSIS Routines */


/*
    Define supported functions for the above features.
*/

#define GR_SPA       "spa"    /* SPA - Switched Path Analysis */
#define GR_ECA       "eca"    /* ECA - External Circuit Analysis */
#define GR_TRK       "trk"    /* TRK - TRK Analysis */
#define GR_NCA       "nca"    /* NCA - Network Controller Analysis */
#define GR_SDA       "sda"    /* SDA - Signal Distributor Analysis */
#define GR_PPA       "ppa"    /* PPA - Pulse Path Analysis */
#define GR_AHA       "aha"    /* AHA - Audit History Analysis */
```

```
/*
    Define return codes for library subroutine, GEN_RNG().
*/

#define GRR_ERR -1      /*
                            An error has been detected.
                        */

#define GRR_ENF  0      /*
                            The requested record has not been
                            found in the generic range file.
                        */


/*
    Define "open bound" or "don't care" indicator.
*/

#define DONT_CARE '-'


/*
    Specify ending sequence and size of each structure element.
*/

#define GR_ENDSEQ  "*0

#define GR_FEATSZ  4
#define GR_FUNCSZ  6
#define GR_GENSZ   6
#define GR_ISSNOSZ 4
#define GR_MXPGM   4
#define GR_MXNAMSZ 12
#define GR_ENDSZ   2


/* Define a union for a record in the GEN_RNG_FIL file */

union GR_REC
{
    char *gr_recptr;     /* Pointer to start of record */
    struct GEN_RNG *gr_rec;    /* Pointer to generic range record */
};



/*
    Define the structure of a "generic range" record.
*/

struct GEN_RNG
{
    char gr_feat[GR_FEATSZ];
                              /*
                                 Feature to which record applies.  See
                                 Note 1.
                              */

    char gr_func[GR_FUNCSZ];
                              /*
                                 Identifies which function of the
                                 feature is to be performed.  See
                                 Note 2.
                              */

    char gr_fgen[GR_GENSZ];
                              /*
                                 Identifies the "from" generic ID
                                 (eg 10, 100, 101).  See Note 3.
                              */

    char gr_fiss[GR_ISSNOSZ];
                              /*
                                 If needed, identifies the "from"
                                 abstract issue number (eg -, 010, 081,
                                 101).  See Note 4.
                              */

    char gr_tgen[GR_GENSZ];
                              /*
                                 Identifies the "to" generic ID
                                 (eg -, 10, 100, 101).  See Note 3.
                              */

    char gr_tiss[GR_ISSNOSZ];
                              /*
                                 If needed, identifies the "to"
                                 abstract issue number (eg -, 010, 081,
                                 101).  See Note 4.
                              */

    char gr_pgms[GR_MXPGM][GR_MXNAMSZ];
                              /*
                                 A list of up to GR_MXPGM routine names
                                 that are to be executed.  See Notes 5
                                 and 6.
                              */

    char gr_end[GR_ENDSZ];
                              /*
                                 Record ending sequence.
                              */
};
```

```
/*
    Declare the value returned by the subroutine, gen_rng().
*/

char *gen_rng();


/*
    Notes:

    1.  Supported features are defined near the beginning of
        this file.

    2.  Supported functions are defined near the beginning of
        this file.

    3.  The elements gr_fgen and gr_tgen specify the "from" and
        "to" generic ID's that are to be used for range checking.
        Gr_fgen specifies the lower bound and must always contain
        either an entire generic ID or the first few digits of the
        generic ID that identify the generic base.  Gr_tgen speci-
        fies the upper bound and may contain a '-' to indicate an
        open upper bound or may contain an entire generic ID or
        the first few digits of the generic ID that identify the
        generic base.  If the value specified for gr_fgen con-
        tains only a generic base, then the value for gr_tgen
        must also contain only a generic base or a '-'.  The
        value specified for either of these elements must be
        left-justified in the appropriate field and padded on
        the right with blanks.

    4.  The elements gr_fiss and gr_tiss specify the "from" and
        "to" abstract issue numbers that are to be used for range
        checking.  These elements should contain the value '-'
        except when it becomes necessary to perform range checking
        on an issue basis rather than just on a generic basis.
        When it does become necessary to perform range checking on
        an issue basis, the element gr_fiss must contain the "from"
        abstract issue number (abstract issue numbers are defined
        in the header file, chldata.h) which specifies the lower
        bound for the range checking.  The element gr_tiss must
        contain the "to" abstract issue number which specifies the
        upper bound for the range checking.  Gr_tiss may contain
        the value '-' as an indication that all abstract issue
        numbers greater than or equal to gr_fiss are to be accepted.
        The value specified for either of these elements must be
        left-justified in the appropriate field and padded on the
        right with blanks.

    5.  Routine names should be of the form:

                aaattbbbbbs
```

where

aaa	contains two or three characters that identify the feature to be performed, such as "rcb" for RC:BUILD and "sca" or "sa" for SCHEDULED ANALYSIS.
tt	is the office type, such as 01 for No. 1 ESS.
bbbbb	contains up to five characters that identify which major phase of the feature is to be performed by this routine, such as "swrf" for SPA reformatting.
s	is a sequence or series code; eg. 'a', 'b', etc., that distinquishes this routine from other routines performing a similiar function for other groups or series of generics and issues.

The program name must be left-justified in the appropriate field, ie. no leading blanks, and all unused characters to the right of the routine name must be filled with blanks.

6. If less than GR_MXPGM routines are required for this feature, then all unused elements of the array, gr_pgms, must be filled with GR_MXNAMSZ blanks. The elements of this array that are needed for each of the supported features, however, must be filled as follows:

FEATURE	GR_PGM[0]	GR_PGM[1]	GR_PGM[2]	GR_PGM[3]
rcb	RCB Main	Unused	Unused	Unused
sca	Analysis Phase	Reformatting Phase	Pre-Analysis Phase	Unused
dca	Analysis Phase	Reformatting Phase	Pre-Analysis Phase	Unused

*/

FILES

/type??/gen_rng	Data File
/usr/include/gen_rng.h	Header File

NAME

gufile - Generic Unit File for Generic Source

DESCRIPTION

The gu file is a control file to be used to reference generic source for generic makes and printing of PR listings.

A gu file defines a version of a PR. Its resident directory is the PR directory with which it is associated. The gu file contains the PR number together with its version number and a collection of pident names. It is this collection of pidents which defines the version of the PR.

The format of a gu file is as follows:

```
     Format                    Example

#PR                   #PR
     <PR number>  <title>        PR-1P137-02   Administrative PR

#OWNER                         #OWNER
     <PR administrator>        jse

#INITIAL PIDENTS               #INITIAL PIDENTS
     <pident name>                ADMINLIB02
     <pident name>
#PIDENTS              #PIDENTS
     <pident name>                MAKE02
     <pident name>                SYSGEN01
                                  TAPEGEN02
#FINAL PIDENTS                 #FINAL PIDENTS
     <pident name>                TRANSFORM02
```

The section names, starting with a pound sign (#), begins in column 1.

The INITIAL PIDENTS section contains zero or more pidents (not au files), one per line. During a make of the PR, the pidents specified here will be made in the order that they are listed and before any of those in the PIDENTS section. For example, a local library of subroutines used by other pidents within the PR would be listed in the INITIAL PIDENTS section. The pidents listed here are those which should be made before any subset of those made in the PIDENTS section.

The PIDENTS section contains zero or more pidents (not au files), one per line. These pidents will be made after those in the INITIAL PIDENTS section. The pidents listed here should be independent of the order in which they are made.

The FINAL PIDENTS section contains zero or more pidents, one per line. The pidents listed here are made after the INITIAL PIDENTS and PIDENTS sections.

The name of the gu file is a concatenation of the PR numbers with the version number and the string ".gu", eg., PR-1P137-02.gu.

The gu file is the responsiblity of the PR administrator who must oversee all changes to the file including the original creation of the file.

To assist the PR administrator in the creation of the gu file a prompt procedure exists. To use it, the PR administrator should perform the following UNIX commands:

```
chdir /pst/jse/aumake
form gu
```

The user will be prompted four times for the necessary information. The user should be familiar with the program _form_ before attempting this procedure.

The gu file is a source file to be maintained like one, ie., a new or changed gu file is placed on genupd with CU's and DU's.

FILES

SEE ALSO

form(I) au(V)

NAME

issue - Issue File

DESCRIPTION

The issue file, _issue_, resides in an appropriate /_type_?? directory for each switching machine (SPCS) that the SCCS supports. This file contains information for each SPCS generic and issue that is supported.

The _issue_ file contains one or more generic-issue messages. Each generic-issue message begins with a message delimiting character whose present value is defined in the header file, _issfil_._h_. Basically, a generic-issue message lists all of the SPCS issues that are officially supported for a specific SPCS generic. Thus, each generic-issue message consists of one generic record followed by one or more issue records that are supported for this generic.

All generic records are fixed size and have the structure **IF_GENREC**, as defined in the header file, _issfil_._h_. All information in a generic record is in ASCII; hence all elements are defined as character strings. Each generic record must be initialized to contain blanks in all elements or unused portions of elements that do not contain data. In general, data in a generic record is left-justified in each element and leading zeroes appear where necessary, such as the generic ID, to insure the required precision.

All issue records are fixed size and have the structure **IF_ISSREC**, as defined in the header file, _issfil_._h_. All information in an issue record is in ASCII; hence all elements are defined as character strings. Each issue record must be initialized to contain blanks in all elements or unused portions of elements that do not contain data. In general, data in an issue record is left-justified in each element.

The following is a listing of the _issfil_._h_ header file.

```
/*
   Header file to define the layout of the issue file that resides
   in the appropriate /type?? directory.
*/


/* Define name of issue file. */

#define ISS_FIL "issue"



/*
    Define valid return codes for the
```

```
        library subroutine GEN_LIST().
    */

    #define GLR_NME 0       /* No more entries exist */
    #define GLR_ERR -1      /* Error detected */



    /*
        Define valid function codes and return codes for the
        library subroutine GEN_NAME().
    */

    #define GNF_GNAM 0      /*  Extract generic name */
    #define GNF_SLANG 1     /*  Extract generic slang name */

    #define GNR_EF 1        /* Entry found */
    #define GNR_ENF 0       /* Requested entry does not exist */
    #define GNR_ERR -1      /* Error detected */



    /*
        Define valid function codes and return codes for the
        library subroutine GET_GEN().
    */

    #define GGF_GNAM 0      /*
                                Use generic name as the generic
                                search key.
                            */
    #define GGF_SLANG 1     /*
                                Use generic slang name as the
                                generic search key.
                            */
    #define GGF_GID 2       /*
                                Use generic ID as the generic
                                search key.
                            */

    #define GGR_ENF 0       /* Requested entry not found */
    #define GGR_ERR -1      /* Error detected */



    /*
        Define valid return code for the
        library subroutine GET_ISS().
    */

    #define GIR_ENF 0       /* Requested entry not found */
```

```
/*
    Define valid return code for the
    library subroutine ISS_LIST().
*/

#define ILR_NME 0       /* No more entries exist */



/*
    Declare types of values returned by library subroutines.
*/

char *gen_list();
char *get_gen();
char *get_iss();
char *iss_list();



/* Define array sizes for structures IF_GENREC and IF_ISSREC */

#define IF_ENDSZ 2
#define IF_GFILLSZ 2
#define IF_GNAMSZ 12
#define IF_SLGSZ 8
#define IF_GIDSZ 6
#define IF_IFILLSZ 6
#define IF_INAMSZ 8
#define IF_IOPSYSZ 8


/* Define ending sequence for each entry in the ISS_FIL file */

#define IF_ENDSEQ "*0


/*
   Define generic-issue message delimiter; this is
   used by the GTMSG subroutine to extract generic-
   issue messages from an "issue" file.  A generic-
   issue message consists of one generic record fol-
   lowed by one or more issue records that are
   associated with the generic record.
*/

#define IF_GIMSG_DLM 03       /* Generic-issue message delimiter */


/* Define a union for an entry in the ISS_FIL file */

union IF_REC
```

```
{
    char *if_recptr;                 /* Pointer to start of record */
    struct IF_GENREC *if_genrec;     /* Pointer to generic record */
    struct IF_ISSREC *if_issrec;     /* Pointer to issue record */
};


/* Define structure of a generic entry in the ISS_FIL file */


struct IF_GENREC
{
    char if_gfill[IF_GFILLSZ];
                          /*
                             Record type and white space.
                             See Note 1.
                          */

    char if_gnam[IF_GNAMSZ];
                          /*
                             Generic name, see Note 2.
                          */

    char if_gslang[IF_SLGSZ];
                          /*
                             Generic slang name, see Note 3.
                          */

    char if_gid[IF_GIDSZ];
                          /*
                             Generic ID, see Note 4.
                          */

    char if_gend[IF_ENDSZ];
                          /*
                             End sequence.
                          */
};


/* Define structure of an issue entry in the ISS_FIL file */

struct IF_ISSREC
{
    char if_ifill[IF_IFILLSZ];
                          /*
                             Record type and white space.
                             See Note 1.
                          */

    char if_inam[IF_INAMSZ];
                          /*
```

```
                                   Issue and point-issue name, see
                                   Note 5.
                                */

        char if_iopsys[IF_IOPSYSZ];
                                /*
                                   Operating system generic issue and
                                   point-issue, see Note 6.
                                */

        char if_iend[IF_ENDSZ];
                                /*
                                   End sequence.
                                */
};


/* Define array sizes for structure GEN_ID (generic ID) */

#define GID_BASESZ 2
#define GID_INFOSZ 4


/* Define structure of generic ID field */

struct GEN_ID
{
        char gid_base[GID_BASESZ];          /* Generic base - see Note 4. */
        char gid_info[GID_INFOSZ];
                                 /*
                                    Generic information - see
                                    Note 4.
                                 */
};


/*
NOTES:
```

1. The fill field contains the generic-issue message delimiter and/or white space. This white space must contain only blank characters (040); tabs are not permitted. This requirement exists so that all entries of the same record type will be fixed length records.

   The fill field for a generic record contains the generic-issue message delimiter, which is a single-character code that must be left-justified in the fill field. The value of this message delimiter is defined elsewhere in this header file. The remainder of the fill field must be padded on the right with blanks.

The fill field for an issue record contains the specified number of blanks.

2. The generic name is the official generic name that has been assigned by the appropriate SPCS development group; this name is not necessarily the PG number. For example, in ESS1A the official name for a generic might be 1AE(C2B4), while in ESS101 the official name might be PG-1H002. The generic name must be left-justified in this field and padded on the right with blanks.

3. The generic slang name is an abbreviated name that is sometimes used in place of the official generic name. Examples are 1E3, 1AE4, etc. The generic slang name, if needed, must be left-justified in this field and padded on the right with blanks. If the slang name is not needed, then this field must be filled with IF_SLGSZ blanks.

4. The generic ID is a two-to-five digit decimal number that uniquely identifies a particular SPCS generic. The format of a generic ID is

   bb[xyz]

   where bb is a two-digit decimal number that identifies a generic base. Examples are:

11	for ESS1, generic 1e3
12	for ESS1, generic 1e4

   xyz is an optional one-to-three digit decimal number that provides additional information that is needed for some SPCS types to uniquely identify a specific generic. Examples are:

   ESS1:
   xyz is a single-digit decimal number that identifies the central processor configuration. A value of 0 identifies those systems that have only a CC; whereas, a value of 1 identifies those systems that have both a CC and a SP. Thus, the generic ID for a 1e3 system having only a CC is 110 and the generic ID for a 1e4 system having both a CC and a SP is 121.

   EPSCS, E911, TN, VSS:
   xyz is a three-digit number that identifies the generic issue and point issue of the operating system that is used in the auxiliary processor.

5. The issue name contains the official issue and point issue that have been assigned by the appropriate development group. Examples are 3.1, 6a.3, and 10c.14, where all

characters to the left of the "." identify the issue and those characters to the right of the "." identify the point issue. The issue name must be left-justified in this field and padded on the right with blanks.

6. The operating system generic issue and point-issue are primarily used for the Auxillary Processor systems that the SCCS supports. It identifies which issue and point-issue of the auxillary processor operating system is being used for the application, such as TN, E911, and VSS.

*/

FILES

/type??/issue	Data File
/usr/include/issfil.h	Header File

NAME

oparm - Office Parameter File

DESCRIPTION

The oparm file in each office directory describes information pertaining to that office. The layout of the oparm file is identical to that of the chldata(V) file with the exception that the "c_name" element contains the "office.channel" name in the chldata file, but only the office name in the oparm file.

FILES

<office directory>/oparm /compool/chldata.h

SEE ALSO

chldata(V)

NAME

passwd - password file

DESCRIPTION

Passwd contains for each user the following information:

name (login name, contains no upper case)
encrypted password
numerical user ID
numerical group ID
initial working directory
program to use as Shell

This is an ASCII file. Each field within each user's entry is separated from the next by a colon. The job and box numbers are separated by a comma. Each user is separated from the next by a new-line. If the password field is null, no password is demanded; if the Shell field is null, the Shell itself is used.

This file resides in directory /etc. Because of the encrypted passwords, it can and does have general read permission and can be used, for example, to map numerical user ID's to names.

SEE ALSO

login(1), crypt(3), passwd(1)

NAME

<patname>.p or <patname>.o - common pattern package pattern file

DESCRIPTION

The compiler creates a pattern in one of the following formats:

> _Standard Form_: This form is found in <_patname_>._p_ files created by the compiler. The pattern contains a header, variable argument information, a pattern and a copy of the definition used to create the pattern (source part). The header contains information as described in the <_ppsubs_._h_> header file. The variable argument information is present only if the pattern is a variable pattern. This information is used by _ppmkpat_(_1L_) when ever the pattern is used. The source part allows standard format patterns to be verified with source output.
>
> _Object Form_: This form is found in <_patname_>._o_ files created by the compiler. The pattern contains a header, a pattern, a relocation map and symbol table. The header contains information as described in the <_ppsubs_._h_> header file which also corresponds to the header of an _a_._out_(_5_) file. This together with the relocation map and symbol table allows the pattern to be loaded into a program by _ld_(_1_) or _cc_(_1_) in the same manner as a **.o** (object) file.

Any time a program other than the compiler makes or modifies a pattern, the pattern is considered to be a "modified form" pattern. Modified form patterns require special information not provided in this document before they can be used.

SEE ALSO

ppdpat(1L), ppmkpat(1L), ppvpat(1L), ld(1), strip(1), cc(1), a.out(5) ppgetpat(3L), ppsccsgp(3L)

NAME

pidentlist - List of pidents of a generic

DESCRIPTION

This file is associated with a pr document and contains, for a specific generic calling out the pr document, the list of pidents associated with the generic.

The file is actually a shell file used in making a generic. The first line contains "$1 $2 $3 $4 $5 $6 \". Each succeeding line has a pident's au file name followed by a backslash (\). The last line is a blank line. The format of the file is fixed by the pgupd (VIII) program, which ignores the first line and reads the succeeding lines, ignoring blank lines. The list of au file must be in alphabetical order for pgupd.

For example,

```
     $1 $2 $3 $4 $5 $6 \
     OPTRBL.au \
     RCTRBL.au \
     TRBLDATA.au \
```

FILES

pgupd(VIII)

SEE ALSO

NAME

PR - PR and PA documents for SCCS Generics

DESCRIPTION

The numbers 1P130-1P209, 5P100-5P101, 5P103-5P152, and 5P300-5P339 are assigned to the No. 2 SCCS project for PG, PR, PD etc. assignments.

Common PR and PA Documents
(1P130 - 1P194)

Gen 2/3 Doc.	Gen 4 Doc.40/70	Gen 5 Doc.40/70	Gen 6 Doc.
PA-1P131-01	PA-1P135-01	PA-1P135-01	
PR-1P131-02	PR-1P131-03/04	PR-1P131-03/04	
Adminstrative			
PR-1P132-02	PR-1P132-03/04	PR-1P132-03/04	
Emergency Change			
PR-1P133-02	PR-1P133-03	PR-1P133-03	
Common Programs			
PR-1P134-01	PR-1P134-02/03	PR-1P134-02/03	
PR-1P135-01	PR-1P135-02	PR-1P135-02	
PR-1P136-01	-----------	-----------	
PR-1P137-01	PR-1P137-02/03	PR-1P137-04/05	
PR-1P138-01	PR-1P138-02	PR-1P138-02	
Installation and			
PR-1P139-01	PR-1P139-02	PR-1P139-03	
Common Patterns			
-----------	PR-1P140-01/02	PR-1P140-03/04	
-----------	PR-1P141-01	PR-1P141-01	
Miscellaneous			
PR-1P154-02	PR-1P154-03	PR-1P154-03	
PR-1P155-02	PR-1P155-03/04	PR-1P155-03/04	
PR-1P156-02	PR-1P156-03	PR-1P156-03	
PR-1P157-02	PR-1P157-02	PR-1P157-02	
-----------	PR-1P158-01	PR-1P158-01	
PR-1P162-02	PR-1P162-03	PR-1P162-03	
Tape Handling			
PR-1P163-02	PR-1P163-03/04	PR-1P163-03/04	
PR-1P164-02	PR-1P164-03	PR-1P164-03	
PR-1P165-02	PR-1P165-03	PR-1P165-03	
PR-1P166-02	PR-1P166-03	PR-1P166-03	
Report Generator			
PR-1P170-02	PR-1P170-03	PR-1P170-03	
Alerter Programs,			
PR-1P171-01	PR-1P171-02	PR-1P171-02	
PR-1P172-02	PR-1P172-03	PR-1P172-03	
Command Inter-			
PR-1P173-01	PR-1P173-02/03	PR-1P173-02/03	
Master Distribut-			
PR-1P174-01	PR-1P174-02	PR-1P174-02	

```
Measurement Pro-
PR-1P175-01    PR-1P175-02/03 PR-1P175-02/03
Schedule Process-
PR-1P178-01    PR-1P178-02    PR-1P178-02
Trouble Reporting
PR-1P179-01    PR-1P179-01    PR-1P179-01
TTY Data Base
PR-1P180-01    PR-1P180-02    PR-1P180-02
TRUMP - Trunk
-----------    -----------    -----------    PR-1P181-01

                          ESS 1 Application Programs
                              (1P195 thru 1P202)

PA-1P195-01    PA-1P199-01    PA-1P199-01
ESS 1 Analysis
PR-1P195-01    PR-1P195-02    PR-1P195-02
ESS 1 Build & Con-
PR-1P196-01    PR-1P196-02    PR-1P196-02
ESS 1 Expansion
PR-1P197-01    PR-1P197-02    PR-1P197-02
PR-1P198-01    PR-1P198-01    PR-1P198-01
ESS 1 Data Base
PR-1P199-01    PR-1P199-02    PR-1P199-02


                         ESS 101 Application Programs
                      (5P060, 5P100 thru 5P107 for ESS 101)
                         (PR-5P102 replaced by PR-5P105)

Gen 2/3 Doc.   Gen 4 Doc.40/70     Gen 5 Doc.40/70

PA-5P060-01    PA-5P100-01    PA-5P100-01
ESS 101 Analysis
PR-5P100-01    PR-5P100-02    PR-5P100-02
ESS 101 Build &
PR-5P101-01    PR-5P101-02    PR-5P101-02
PR-5P103-01    PR-5P103-02    PR-5P103-02
ESS 101 Data Base
PR-5P104-01    PR-5P104-02    PR-5P104-02
ESS 101 Expansion
PR-5P105-01    PR-5P105-02    PR-5P105-02


                          ESS 2 Application Programs
                          (5P108 thru 5P114 for ESS2)

Gen 3 Doc.     Gen 4 Doc.40/70     Gen 4 Doc.40/70

PA-1P196-01    PA-5P108-01    PA-5P108-01
ESS 2 Analysis
-----------    PR-5P108-01    PR-5P108-01
```

ESS 2 Build and		
PR-5P109-01	PR-5P109-02	PR-5P109-02
ESS 2 Expansion		
PR-5P110-01	PR-5P110-02	PR-5P110-02
PR-5P111-01	PR-5P111-01	PR-5P111-01
ESS 2 Data Base		
PR-5P112-01	PR-5P112-02	PR-5P112-02

TSPS Application Programs
(5P070, 5P115 thru 5P123 for TSPS)

<u>Gen 2/3 Doc</u>.	<u>Gen 4 Doc</u>.<u>40/70</u>	<u>Gen 5 Doc</u>.<u>40/70</u>
PA-5P070-01	PA-5P115-01	PA-5P115-01
TSPS Analysis		
PR-5P115-01	PR-5P115-02	PR-5P115-02
TSPS Buils & Con-		
PR-5P116-01	PR-5P116-02	PR-5P116-02
PR-5P117-01	PR-5P117-02	PR-5P117-02
TSPS Expansion		
PR-5P118-01	PR-5P118-02	PR-5P118-02
PR-5P119-01	PR-5P119-02	PR-5P119-02
TSPS Data Base		
PR-5P120-01	PR-5P120-02	PR-5P120-02

*includes thresholding programs

ESS 2B Application Programs
(5P124 thru 5P130 for ESS2B)

<u>Gen 3 Doc</u>.	<u>Gen 4 Doc</u>.<u>40/70</u>	<u>Gen 5 Doc</u>.<u>40/70</u>
PA-1P197-01	PA-5P124-01	PA-5P124-01
ESS 2B Analysis		
-----------	PR-5P124-01	PR-5P124-01
ESS 2B Build and		
PR-5P125-01	PR-5P125-02	PR-5P125-02
ESS 2B Expansion		
PR-5P126-01	PR-5P126-02	PR-5P126-02
PR-5P127-01	PR-5P127-01	PR-5P127-01
ESS 2B Data Base		
PR-5P128-01	PR-5P128-02	PR-5P128-02

ESS 3 Application Programs
(5P131 thru 5P137 for ESS3)

<u>Gen 3 Doc</u>.	<u>Gen 4 Doc</u>.<u>40/70</u>	<u>Gen 5 Doc</u>.<u>40/70</u>
PA-1P198-01	PA-5P131-01	PA-5P131-01

ESS 3 Build and		
PR-5P132-01	PR-5P132-01	PR-5P132-01
ESS 3 Expansion		
PR-5P133-01	PR-5P133-01	PR-5P133-01
PR-5P134-01	PR-5P134-02	PR-5P134-02
ESS 3 Data Base		
PR-5P135-01	PR-5P135-02	PR-5P135-02

AMARS Application Programs
(5P138 thru 5P144 for AMARS)

Gen 2/3 Doc.	Gen 4 Doc.40/70	Gen 5 Doc.40/70
-----------	PA-5P138-01	PA-5P138-01
AMARS Data Base		
PR-5P138-01	PR-5P138-02	PR-5P138-02
PR-5P139-01	PR-5p139-02	PR-5P139-02

ESS 1A Application Programs
(5P145 thru 5P151 for ESS 1A)

Gen 2/3 Doc.	Gen 4 Doc.40/70	Gen 5 Doc.40/70
-----------	PA-5P145-01	PA-5P145-01
ESS 1A Analysis		
-----------	PR-5P145-01	PR-5P145-02
ESS 1A Build and		
-----------	PR-5P146-01	PR-5P146-01
ESS 1A Expansion		
-----------	PR-5P147-01	PR-5P147-01
-----------	PR-5P148-01	PR-5P148-01
ESS 1A Data Base		
-----------	PR-5P149-01	PR-5P149-01
ESS 1A Display		
-----------	PR-5P150-01	PR-5P150-01

ESS 4 Application Programs
(1P203 thru 1P209 for ESS 4)

Gen 2/3 Doc.	Gen 4 Doc.40/70	Gen 5 Doc.40/70
-----------	-----------	PA-1P203-01
ESS 4 Analysis		
-----------	PR-1P203-01	PR-1P203-01
ESS 4 Build and		
-----------	PR-1P204-01	PR-1P204-01
ESS 4 Expansion		
-----------	PR-1P205-01	PR-1P205-01
-----------	PR-1P206-01	PR-1P206-01

ESS 4 Data Base		
-----------	PR-1P207-01	PR-1P207-01
Ess4 and 1A Common		
-----------	PR-1P208-01	PR-1P208-02

TN Application Program
(5P300-5P305) (type10)

Gen 2/3 Doc.	Gen 4 Doc.40/70	Gen 5 Doc.40/70
-----------	-----------	PA-5P300-01
-----------	-----------	PR-5P300-01
TN Build and		
-----------	-----------	PR-5P301-01
-----------	-----------	PR-5P302-01
-----------	-----------	PR-5P303-01
TN Data Base Pgms		
-----------	-----------	PR-5P304-01

PDSP (EPSCS/E911) Application Program
(5P306-5P311) (EPSCS type11) (E911 type15)

Gen 2/3 Doc.	Gen 4 Doc.40/70	Gen 5 Doc.40/70
-----------	-----------	PA-5P306-01
-----------	-----------	PA-5P307-01
-----------	-----------	PR-5P306-01
PDSP Build and		
-----------	-----------	PR-5P307-01
PDSP Expansion		
-----------	-----------	PR-5P308-01
-----------	-----------	PR-5P309-01
PDSP Data Base		
-----------	-----------	PR-5P310-01

VSS Application Program
(5P312-5P317) (type12)

Gen 2/3 Doc.	Gen 4 Doc.40/70	Gen 5 Doc.40/70
-----------	-----------	PA-5P312-01
-----------	-----------	PR-5P312-01
VSS Build and		
-----------	-----------	PR-5P313-01
VSS Expansion		
-----------	-----------	PR-5P314-01
-----------	-----------	PR-5P315-01
VSS Data Base		
-----------	-----------	PR-5P316-01

ACS Application Program
(5P318-5P323) (type13)

Gen 2/3 Doc.	Gen 4 Doc.40/70	Gen 5 Doc.40/70
-----------	-----------	PA-5P318-01
-----------	-----------	PR-5P318-01
ACS Build and		
-----------	-----------	PR-5P319-01
ACS Expansion		
-----------	-----------	PR-5P320-01
-----------	-----------	PR-5P321-01
ACS Data Base		
-----------	-----------	PR-5P322-01

AIS Application Program
(5P324-5P329) (type14)

Gen 2/3 Doc.	Gen 4 Doc.40/70	Gen 5 Doc.40/70
-----------	-----------	PA-5P324-01
-----------	-----------	PR-5P324-01
AIS Build and		
-----------	-----------	PR-5P325-01
AIS Expansion		
-----------	-----------	PR-5P326-01
-----------	-----------	PR-5P327-01
AIS Data Base		
-----------	-----------	PR-5P328-01

EOS Application Program
(5P330-5P334)

Gen 2/3 Doc.	Gen 4 Doc.40/70	Gen 5 Doc.40/70
PA Manual		
EOS Build and		
-----------	-----------	PR-5P330-01
EOS Expansion		
-----------	-----------	PR-5P331-01
-----------	-----------	PR-5P332-01
EOS Data Base		
-----------	-----------	PR-5P333-01

SCCS Application Program
(5P335-5P339) (type 00)

Gen 2/3 Doc.	Gen 4 Doc.40/70	Gen 5 Doc.40/70
-----------	-----------	PA-5P335-01

SCCS Data Base		
-----------	-----------	PR-5P335-01
-----------	-----------	PR-5P336-01
-----------	-----------	PR-5P337-01
-----------	-----------	PR-5P338-01
-----------	-----------	PR-5P339-01

The following pr documents are for UNIX commands, liba, libc, and the UNIX operating system.

Gen 2/3 Doc.	Gen 4 Doc.40/70	Gen 5 Doc.40/70
PR-1C304-11	PR-1C304-12	PR-1C304-12
PR-1C306-01	PR-1C306-12/13	PR-1C306-12/13
Desk Calculator		
PR-1C307-01	PR-1C307-11	PR-1C307-11
PR-1C310-11	PR-1C310-12	PR-1C310-12
Dec/mag Tape Man-		
PR-1C311-11	PR-1C311-12	PR-1C311-12
Floating Point		
PR-1C312-11	PR-1C312-11	PR-1C312-11
PR-1C313-11	PR-1C313-12	PR-1C313-12
PR-1C314-11	PR-1C314-12	PR-1C314-12
PR-1C315-11	PR-1C315-12	PR-1C315-12
PR-1C316-11	PR-1C316-12	PR-1C316-12
PR-1C317-11	PR-1C317-12	PR-1C317-12
PR-1C318-11	PR-1C318-12	PR-1C318-12
UNIX Assembler		
PR-1C319-11	PR-1C319-12	PR-1C319-12
UNIX Assembler		
PR-1C320-11	PR-1C320-12	PR-1C320-12
UNIX C Subroutines		
PR-1C321-11	PR-1C321-12/13	PR-1C321-12/13
UNIX C Subroutines		
PR-1C322-11	PR-1C322-12	PR-1C322-12
UNIX C Subroutines		
PR-1C323-11	PR-1C323-12/13	PR-1C323-12/13
UNIX C Subroutines		
PR-1C324-11	PR-1C324-12	PR-1C324-12
UNIX C Subroutines		
PR-1C325-11	PR-1C325-12	PR-1C325-12
UNIX C Subroutines		
PR-1C326-11	PR-1C326-12	PR-1C326-12
UNIX C Subroutines		
PR-1C327-11	PR-1C327-12/13	PR-1C327-12/13
PR-1C328-11	PR-1C328-12	PR-1C328-12
UNIX - Definitions		
PR-1P143-02	PR-1P143-03/04	PR-1P143-03/05
UNIX - System		
PR-1P144-02	PR-1P144-03/04	PR-1P144-03/04
PR-1P145-02	PR-1P145-03/04	PR-1P145-03/04

FILES

SEE ALSO

NAME

prindex - PR document index page

DESCRIPTION

The prindex file contains information about a PR document associated with a PG - generic. The prindex file is divided into three ordered sections. They are:

#PRDOC

The #PRDOC section contains the PR document number and its title. The version, shown below in the example, is optional.

#PIDENT

The #PIDENT section contains the pidents associated with the PR document. There is one pident per line; the name of the pident is listed first followed by the issue sequence of the pident and then the title of the pident. The issue sequence is a string of issues of the pident, each issue separated by commas. If the first character of a line in this section is an asterisk (*) then the pident associated with the line no longer exists. This way a record of all pidents (deleted or existing) associated with a pr document can be kept.

#PRINT

The #PRINT section is an optional section which contains pident names. This field can be used to list changed pidents of a generic. The command, plistp can then be used to print only the changed pidents for a letter issue release of a generic. The pidents are listed one per line.

An example of a prindex file:

```
#PRDOC
     PR-1P177-01    1,2,3     Trouble Reporting Programs
#PIDENT
     OPTRBL          1,1,1     Output Trouble Program
     RCTRBL          1,2,3     Recent Change Trouble Program
*    TRBLDATA        1,1       Trouble Reporting Data Base
#PRINT
     RCTRBL
```

FILES

SEE ALSO

plistp(I), au file(V), secprt(I)