z/OS UNIX Basic Security

Julie Bergh

berghju@gmail.com

- **Overview**
- BPXPRMxx
- BPXISEC1
- Next Steps

Introduction to UNIX

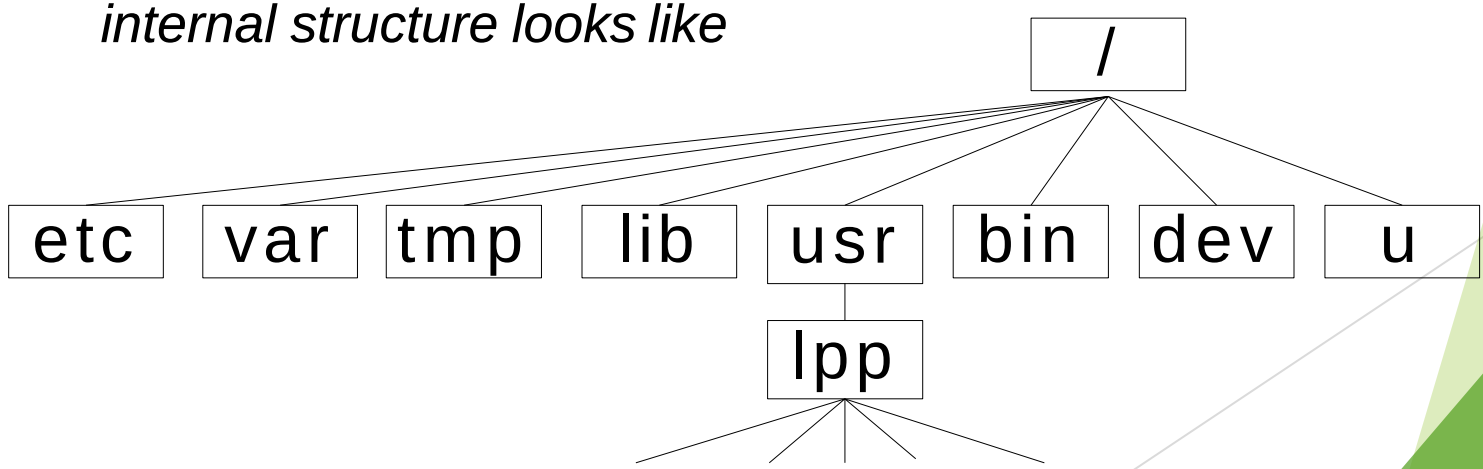
- Kernel
  - The heart of the system – provides UNIX services to it‟s callers (“system calls”)
  - In z/OS part of the Basic Control Program (BCP)
- Shell
  - An interface between a User and the Kernel
  - Accepts, interprets, and executes your commands
- DAEMONS - Processes that run in background i.e. Started Tasks
- File system
  - Hierarchical directory structure for storing data (in “files”)
  - A whole file system in z/OS UNIX is stored in one or more Data Sets (HFS or zFS)

Interfacing with z/OS UNIX

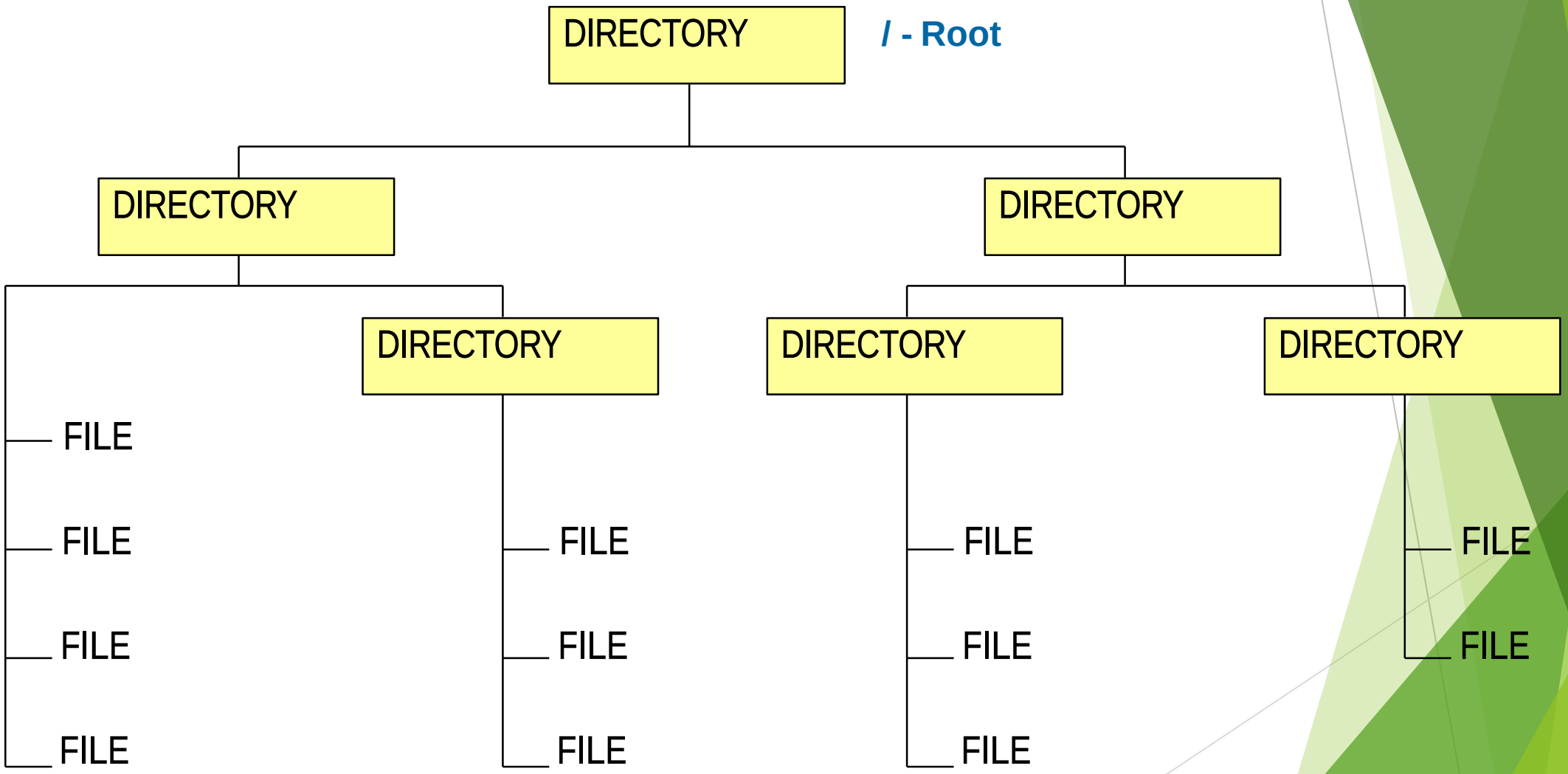
- UNIX terminal (VT100, VT220, xterm)
  - Interactive work via telnet or ssh
  - This is the standard (and typically only way) on other platforms
- 3270 (TSO OMVS, ISPF Shell)
  - OMVS
    - Type in a command or two and read the output
  - ISHELL
    - The MVS-like way of doing things (through ISPF panels)
- Batch
  - UNIX services (APIs) in application programs
    - TCP/IP, Java, Web servers, Application servers
  - UNIX tools to process datasets (text processing tools)

z/OS UNIX file system

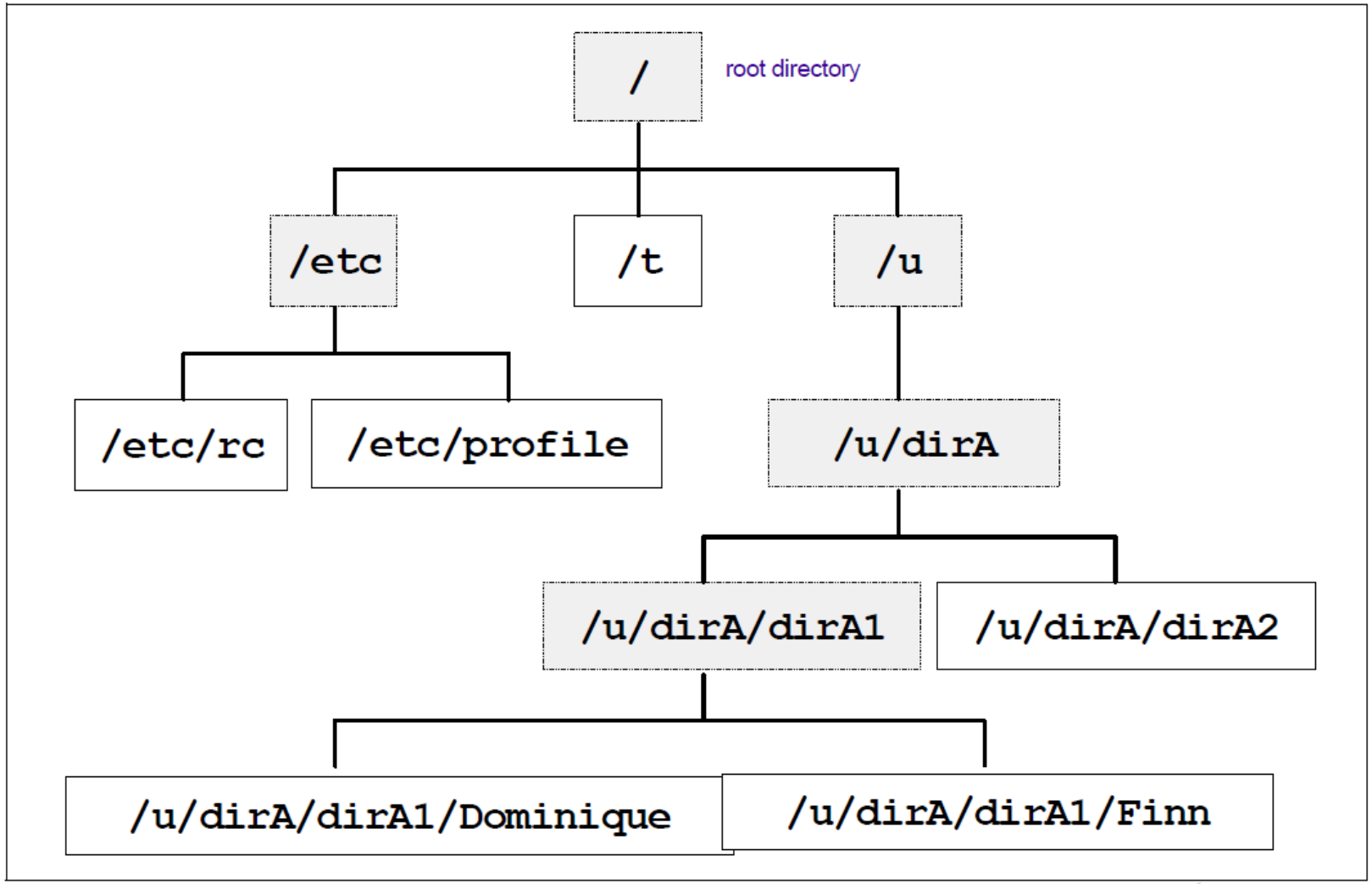
- UNIX file system is a hierarchical directory sturcture
  - Starts with a root "/"
  - Every file or a directory contained in its "parent" directory ".."
    - Parent directory of root is root
    - Files are just streams of bytes
      - *No internal structure from the operating system perspective*
      - *Application working with a file has to understand what the internal structure looks like*



File System


DIRECTORY
/ - Root
DIRECTORY
DIRECTORY
DIRECTORY
DIRECTORY
DIRECTORY
FILE
FILE
FILE
FILE
FILE
FILE
FILE
FILE
FILE
FILE
FILE
FILE

File System


/
root directory
/etc
/t
/u
/etc/rc
/etc/profile
/u/dirA
/u/dirA/dirA1
/u/dirA/dirA2
/u/dirA/dirA1/Dominique
/u/dirA/dirA1/Finn

- Overview
- **BPXPRMxx**
- BPXISEC1
- Next Steps

SYS1.PARMLIB(BPXPRMxx)

```
/*****************************************************************/ 01200000
/*                                                               */ 01210000
/*   Specify the maximum number of processes that z/OS UNIX      */ 01220000
/*   will allow to be active concurrently.                       */ 01230000
/*                                                               */ 01240000
/*   Notes:                                                      */ 01250000
/*                                                               */ 01260000
/*   1. Minimum allowable value is 5.                            */ 01270000
/*   2. Maximum allowable value is 32767.                        */ 01280000
/*   3. If this parameter is not provided, the system default    */ 01290000
/*      value for this parameter is 900.                         */ 01300000
/*                                                               */ 01310000
/*****************************************************************/ 01320000
  MAXPROCSYS(900)                 /* System will allow at most 900    01330000
                                     processes to be active           01340000
                                     concurrently             @P9C*/ 01350000
                                                                     01360000
```

SYS1.PARMLIB(BPXPRMxx)

```
/*****************************************************************/ 01370000
/*                                                               */ 01380000
/*   Specify the maximum number of processes that a single user  */ 01390000
/*   (that is, with the same UID) is allowed to have concurrently*/ 01400000
/*   active regardless of origin.                                */ 01410000
/*                                                               */ 01420000
/*   Notes:                                                      */ 01430000
/*                                                               */ 01440000
/*   1. This parameter is the same as the Child_Max variable     */ 01450000
/*      defined in POSIX 1003.1.                                 */ 01460000
/*   2. Minimum allowable value is 3.                            */ 01470000
/*   3. Maximum allowable value is 32767.                        */ 01480000
/*   4. If this parameter is not provided, the system default    */ 01490000
/*      value for this parameter is 25.                          */ 01500000
/*                                                               */ 01510000
/*****************************************************************/ 01520000
  MAXPROCUSER(25)                 /* Allow each user (same UID) to  01530000
                                     have at most 25 concurrent     01540000
                                     processes active            */ 01550000
                                                                    01560000
```

SYS1.PARMLIB(BPXPRMxx)

```
/*   Notes:                                                        */ 05580000
/*                                                                 */ 05590000
/*   1. There can only be one ROOT statement.                      */ 05600000
/*   2. FILESYSTEM can be up to 44 characters.                     */ 05610000
/*      It must be entered as a quoted string.                     */ 05620000
/*   3. DDNAME is the name of a DD statement in a UNIX System      */ 05630000
/*      Services PROC.  It can be up to 8 characters.              */ 05640000
/*   4. FILESYSTEM and DDNAME are mutually exclusive.              */ 05650000
/*      Exactly one of them must be specified.                     */ 05660000
/*   5. TYPE is required and can be up to 8 characters.            */ 05670000
/*      This matches the TYPE specified in a FILESYSTYPE statement.*/ 05680000
/*   6. PARM can be up to  500 characters.                         */ 05690000
/*      It must be entered as a quoted string.  It can be entered  */ 05700000
/*      in mixed case, as required by the physical file system,    */ 05710000
/*      e.g. PARM ('80').                                          */ 05720000
/*      For specific information about values to specify for PARM  */ 05730000
/*      refer to either 'z/OS MVS Initialization and Tuning        */ 05740000
/*      Reference' or to documentation for the specific PFS.       */ 05750000
/*   7. MODE is either 'READ' or 'RDWR'.  Default for MODE is      */ 05760000
/*      'RDWR' (read/write).                                       */ 05770000
/*   8. The specific parameters and values for the parameters are  */ 05780000
/*      file system dependent.  Refer to 'Z/OS UNIX System Services*/ 05790000
/*      Planning' for the file system that owns the root file set. */ 05800000
/*   9. SETUID or NOSETUID can be specified to support or ignore   */ 05810000
/*      the setuid() and setgid() mode bits on an executable file. */ 05820000
/*      The default is SETUID.                                     */ 05830000
/*  10. If no ROOT statement is found, then the temporary file     */ 05840000
/*      system will be started if it has not been specified, and   */ 05850000
/*      a TFS root will be mounted.  This root is empty initially. */ 05860000
/*                                                                 */ 05870000
/*****************************************************************/ 05880000
/*ROOT FILESYSTEM('OMVS.ROOT')*/                                     05890000
                                  /* Either FILESYSTEM or DDNAME must 05900000
                                     be specified, but not both.     05910000
                                     FILESYSTEM must be entered in   05920000
                                     quotes.                      */ 05930000
/*         TYPE(ZFS)            *//* Type of File system       @DJC*/ 05940000
/*         MODE(RDWR)           *//* (Optional) Can be READ or RDWR.  05950000
/*                                  Default = RDWR               */ 05960000
/*         MKDIR('...')         *//* Mountpoint to be created  @DFA*/ 05970000
                                                                    05980000
```

SYS1.PARMLIB(BPXPRMxx)

```
/*****************************************************************/ 10870000
/*                                                               */ 10880000
/*   SUPERUSER is a 1 to 8-character name that must conform to the  */ 10890000
/*   restrictions for an MVS user ID.  This user ID is assigned     */ 10900000
/*   to shell users when they enter the su command.  This user ID   */ 10910000
/*   should be defined to the security product and have a UID of 0  */ 10920000
/*   assigned to it.                                             */ 10930000
/*   The default is SUPERUSER(BPXROOT).                          */ 10940000
/*                                                               */ 10950000
/*****************************************************************/ 10960000
  SUPERUSER(BPXROOT)                                               10970000
                                                                   10980000
```

SYS1.PARMLIB(BPXPRMxx)

```
/*****************************************************************/ 10990000
/*                                                               */ 11000000
/*   TTYGROUP is a 1 to 8-character name that must conform to the*/ 11010000
/*   restrictions for an MVS group name.  Slave pseudoterminals  */ 11020000
/*   (ptys) and OCS rtys are given this group name when they are */ 11030000
/*   first opened.  The name is used by certain setgid programs, */ 11040000
/*   such as talk and write, when attempting to write to another */ 11050000
/*   user's pty or rty.  This group name should be defined to the*/ 11060000
/*   security product and have a unique GID.  No users should be */ 11070000
/*   connected to this group.                                    */ 11080000
/*                                                               */ 11090000
/*   The default is TTYGROUP(TTY).                               */ 11100000
/*                                                               */ 11110000
/*****************************************************************/ 11120000
  TTYGROUP(TTY)                                                    11130000
                                                                   11140000
```

SYS1.PARMLIB(BPXPRMxx)

```
/*********************************************************************/ 11150000
/*                                                                   */ 11160000
/*   A 1-8 character name of started procedure JCL initializing      */ 11170000
/*   the z/OS UNIX kernel.                                           */ 11180000
/*   Default: OMVS                                                   */ 11190000
/*                                                                   */ 11200000
/*********************************************************************/ 11210000
  STARTUP_PROC(OMVS)                                                   11220000
                                                                       11230000
```

SYS1.PARMLIB(BPXPRMxx)

```
/*********************************************************************/ 12510000
/*                                                                   */ 12520000
/* RESOLVER_PROC is used to specify how the resolver address space   */ 12530000
/* is processed during Unix System Services initialization.          */ 12540000
/* The resolver address space is used by Tcp/Ip applications         */ 12550000
/* for name-to-address or address-to-name resolution.                */ 12560000
/* In order to create a resolver address space, a system must be     */ 12570000
/* configured with an AF_INET or AF_INET6 domain.                    */ 12580000
/*                                                                   */ 12590000
/*  RESOLVER_PROC(procname|DEFAULT|NONE)                             */ 12600000
/*                                                                   */ 12610000
/*     procname - The name of the address space for the resolver.    */ 12620000
/*                In this case, this is the name of the address      */ 12630000
/*                space as well as the procedure member name         */ 12640000
/*                in PROCLIB.  procname is 1 to 8 characters long.   */ 12650000
/*                                                                   */ 12660000
/*     DEFAULT -  An address space with the name RESOLVER will       */ 12670000
/*                be started.  This is the same result that will     */ 12680000
/*                occur if the RESOLVER_PROC statement is not        */ 12690000
/*                specified in the BPXPRMxx profile.                 */ 12700000
/*                                                                   */ 12710000
/*     NONE    -  Specifies that a RESOLVER address space is         */ 12720000
/*                not to be started.                                 */ 12730000
/*                                                                @DAA*/ 12740000
/*********************************************************************/ 12750000
RESOLVER_PROC(DEFAULT)                                                  12760000
                                                                        12770000
```

SYS1.PARMLIB(BPXPRMxx)

```
/*****************************************************************/ 14290000
/*  UMASK(nnnn)                                                  */ 14300000
/*                                                               */ 14300100
/* Specifies default file creation permission-code mask (umask)  */ 14301000
/* to the given mode for all users of z/OS UNIX.                 */ 14310000
/* nnnn is a 3 to 4 digit octal number, which specifies the      */ 14320000
/* permission settings of files/directories created by users or  */ 14330000
/* processes that are not to be allowed.                         */ 14340000
/* nnnn can also equal to NONE, which will indicate no system umask */ 14350000
/* default has been specified.                                   */ 14360000
/*                                                               */ 14370000
/* Default: If a UMASK value is not specified in BPXPRMxx, the   */ 14380000
/*          default will be NONE.                                */ 14390000
/*                                                               */ 14400000
/* Note: Use this setting to better control what files/directories  */ 14410000
/*       users on your system can access.                        */ 14420000
/*       For example, specifying a UMASK value of 007 will ensure   */ 14421000
/*       that only the creating file/directory owner and users in   */ 14422000
/*       the associated group have access to a newly created     */ 14423000
/*       file/directory.                                         */ 14424000
/*****************************************************@03A*/ 14430000
/* UMASK(007) */                                                  14440000
```

- Overview
- BPXPRMxx
- **BPXISEC1**
- Next Steps

SYS1.SAMPLIB(BPXISEC1)

```
/*                        L E G E N D                                */
/*                  ====================================             */
/*                                                                   */
/*  OMVSKERN - refers to the user ID which has been assigned as      */
/*             the owning user for the OMVS and BPXOINIT cataloged   */
/*             procedures. In the z/OS UNIX Planning book we         */
/*             commonly refer to this also as the kernel user ID.    */
/*                                                                   */
/*  OMVSGRP  - refers to the group ID which has been assigned as     */
/*             the owning group for the OMVS and BPXOINIT cataloged  */
/*             procedures. In the z/OS UNIX Planning book we         */
/*             commonly refer to this also as the kernel group ID.   */
/*                                                                   */
/*  Both OMVSKERN and OMVSGRP names can be changed to meet your      */
/*  site naming conventions throughout this CLIST.                   */
/*                                                                   */
/*******************************************************************/
```

SYS1.SAMPLIB(BPXISEC1)

```
/*************************************************************/
/*  Block 1 - Defining owning user ID and group ID           */
/*                                                           */
/* To define the OMVSKERN and OMVSGRP entries.               */
/*                                                           */
/*  ATTENTION:                                               */
/*  The OMVSKERN user ID is defined as a PROTECTED user ID by*/
/*  using the NOPASSWORD operand.                            */
/*  If you share the RACF database, protected user IDs may be*/
/*  used to attempt logon from systems running OS/390 releases*/
/*  prior to OS/390 Version 2 Release 8.                     */
/*  This may result in protected user IDs being revoked through*/
/*  incorrect password attempts from downlevel systems.      */
/*  To avoid this, you should make sure that OS/390 Version 2*/
/*  Release 8 or higher is installed on all shared systems before*/
/*  defining protected user IDs.                             */
/*                                                           */
/* Customizable entries:                                     */
/* ====================                                      */
/*  a) x - refers to a unique GID number between 0 - 2147483647*/
/*     Do not permit the kernel group id, OMVSGRP, to any MVS*/
/*     resources, unless programs you start using /etc/rc need to*/
/*     be permitted to these resources.                      */
/*                                                           */
/*************************************************************/

/* ADDGROUP OMVSGRP OMVS(GID(x))                             */
/* ADDUSER OMVSKERN DFLTGRP(OMVSGRP) OMVS(UID(0) HOME('/') + */
/*   PROGRAM('/bin/sh')) NOPASSWORD                          */
```

SYS1.SAMPLIB(BPXISEC1)

```
/*************************************************************/
/*  Block 2 - OMVS Cataloged Procedure                       */
/*                                                           */
/* You must define OMVS to the RACF STARTED FACILITY.        */
/* Customers who want to use the started procedure table (ICHRIN03) */
/* instead, should refer to the z/OS UNIX Planning book.     */
/*                                                           */
/* Customizable entries:                                     */
/* ====================                                      */
/*  a) If you decide to make OMVS not a trusted procedure, change   */
/*     TRUSTED(YES) to TRUSTED(NO) below.                    */
/*                                                           */
/*************************************************************/

/* SETROPTS GENERIC(STARTED)                                 */
/* RDEFINE STARTED OMVS.* STDATA(USER(OMVSKERN) GROUP(OMVSGRP) +    */
/*   TRUSTED(YES))                                           */
/* SETROPTS CLASSACT(STARTED) RACLIST(STARTED)               */
/* SETROPTS RACLIST(STARTED) REFRESH                         */

```

SYS1.SAMPLIB(BPXISEC1)

```
/*************************************************************/
/*  Block 3 - BPXOINIT Cataloged Procedure                   */
/*                                                           */
/* You must define BPXOINIT to the RACF STARTED FACILITY.    */
/* Customers who want to use the started procedure table (ICHRIN03) */
/* instead, should refer to the z/OS UNIX Planning book.     */
/*                                                           */
/* Do not make BPXOINIT a trusted procedure.                 */
/*                                                           */
/*************************************************************/

/* SETROPTS GENERIC(STARTED)                                 */
/* RDEFINE STARTED BPXOINIT.* STDATA(USER(OMVSKERN) GROUP(OMVSGRP) +*/
/*   TRUSTED(NO))                                            */
/* SETROPTS CLASSACT(STARTED) RACLIST(STARTED)               */
/* SETROPTS RACLIST(STARTED) REFRESH                         */

/*************************************************************/
```

SYS1.SAMPLIB(BPXISEC1)

```
/*************************************************************/
/*  Block 4 - Defining user IDs as UNIX superusers           */
/*                                                           */
/* A superuser is (in its simplest terms) a user ID with a   */
/* UID of 0. This is usually needed for users who run SMP/E jobs.  */
/*                                                           */
/* Customizable entries:                                     */
/* ====================                                      */
/*  a) Change ALTUSER to ADDUSER, if this user ID is new to the  */
/*     database.                                             */
/*  b) xxxxxxxx - refers to the user ID.                     */
/*  c) Ensure this user ID's default group has a GID.        */
/*                                                           */
/*************************************************************/

/* ALTUSER xxxxxxxx OMVS(UID(0) HOME('/') PROGRAM('/bin/sh'))    */

```

SYS1.PARMLIB(BPXPRMxx)

```
/*****************************************************************/ 10870000
/*                                                               */ 10880000
/*   SUPERUSER is a 1 to 8-character name that must conform to the  */ 10890000
/*   restrictions for an MVS user ID.  This user ID is assigned     */ 10900000
/*   to shell users when they enter the su command.  This user ID   */ 10910000
/*   should be defined to the security product and have a UID of 0  */ 10920000
/*   assigned to it.                                             */ 10930000
/*   The default is SUPERUSER(BPXROOT).                          */ 10940000
/*                                                               */ 10950000
/*****************************************************************/ 10960000
  SUPERUSER(BPXROOT)                                               10970000
                                                                   10980000
```

SYS1.SAMPLIB(BPXISEC1)

```
/*****************************************************************/
/*  Block 5 - Defining regular users and groups                  */
/*                                                               */
/* This example illustrates how to define a regular user and a   */
/* group. Any non-zero UID user is a regular user.               */
/*                                                               */
/* Customizable entries:                                         */
/* =====================                                         */
/*  a) Change ALTGROUP to ADDGROUP, if this group ID is new to   */
/*     the database.                                             */
/*  b) YYYYYYYY - refers to the group ID.                        */
/*  c) W - refers to numeric value between 0 - 2147483647.       */
/*                                                               */
/*  d) Change ALTUSER to ADDUSER, if this user ID is new to the  */
/*     database. When adding a new user, ensure you specify      */
/*     DFLTGRP(YYYYYYYY).                                        */
/*  e) XXXXXXXX - refers to the user ID.                         */
/*  f) Z - refers to numeric value between 0 - 2147483647.       */
/*  g) xxxxxxxx - refers to the user ID in lowercase.            */
/*     In UNIX, users' HOME directory is usually /u/userid.      */
/*                                                               */
/*****************************************************************/

/* ALTGROUP YYYYYYYY OMVS(GID(W))                                */

/* ALTUSER XXXXXXXX OMVS(UID(Z) HOME('/u/xxxxxxxx') +            */
/*   PROGRAM('/bin/sh'))                                         */
```

SYS1.SAMPLIB(BPXISEC1)

```
/*****************************************************************/
/*  Block 6 - Defining BPXROOT user                              */
/*                                                               */
/*  In order for daemon processes to be able to invoke setuid() for */
/*  superusers, define a superuser with a user ID of BPXROOT on all */
/*  systems.                                                     */
/*                                                               */
/*  On the SUPERUSER statement in the BPXPRMxx parmlib member,   */
/*  specify the user ID that the kernel will use when you need a */
/*  user ID for UID(0).                                          */
/*                                                               */
/*  ATTENTION:                                                   */
/*  The BPXROOT user ID is defined as a PROTECTED user ID by     */
/*  using the NOPASSWORD operand.                                */
/*  If you share the RACF database, protected user IDs may be    */
/*  used to attempt logon from systems running OS/390 releases   */
/*  prior to OS/390 Version 2 Release 8.                         */
/*  This may result in protected user IDs being revoked through  */
/*  incorrect password attempts from downlevel systems.          */
/*  To avoid this, you should make sure that OS/390 Version 2    */
/*  Release 8 or higher is installed on all shared systems before */
/*  defining protected user IDs.                                 */
/*                                                               */
/*****************************************************************/

/* ADDUSER BPXROOT DFLTGRP(OMVSGRP) +                            */
/* OMVS(UID(0) HOME('/') PROGRAM('/bin/sh')) NOPASSWORD          */
```

SYS1.PARMLIB(BPXPRMxx)

```
/*****************************************************************/ 10870000
/*                                                               */ 10880000
/*   SUPERUSER is a 1 to 8-character name that must conform to the  */ 10890000
/*   restrictions for an MVS user ID.  This user ID is assigned     */ 10900000
/*   to shell users when they enter the su command.  This user ID   */ 10910000
/*   should be defined to the security product and have a UID of 0  */ 10920000
/*   assigned to it.                                             */ 10930000
/*   The default is SUPERUSER(BPXROOT).                          */ 10940000
/*                                                               */ 10950000
/*****************************************************************/ 10960000
  SUPERUSER(BPXROOT)                                               10970000
                                                                   10980000
```

SYS1.SAMPLIB(BPXISEC1)

```
/*********************************************************************/
/*  Block 7 - Defining TTY group                                     */
/*                                                                   */
/*  In order to use the talk, write and mesg utilities, you will     */
/*  need to define the TTY group.                                    */
/*                                                                   */
/* Customizable entries:                                             */
/* =====================                                             */
/*  a) xxxxxxx - A unique GID and no users should be connected       */
/*     to this group.                                                */
/*  b) TTY - If this name doesn't conform to your site's naming      */
/*     conventions, then use a different group name and place        */
/*     this name in the TTYGROUP statement in the BPXPRMxx           */
/*     parmlib member.                                               */
/*                                                                   */
/*********************************************************************/

/* ADDGROUP TTY OMVS(GID(xxxxxxx))                                   */

```

SYS1.PARMLIB(BPXPRMxx)

```
/*****************************************************************/ 10990000
/*                                                               */ 11000000
/*   TTYGROUP is a 1 to 8-character name that must conform to the*/ 11010000
/*   restrictions for an MVS group name.  Slave pseudoterminals  */ 11020000
/*   (ptys) and OCS rtys are given this group name when they are */ 11030000
/*   first opened.  The name is used by certain setgid programs, */ 11040000
/*   such as talk and write, when attempting to write to another */ 11050000
/*   user's pty or rty.  This group name should be defined to the*/ 11060000
/*   security product and have a unique GID.  No users should be */ 11070000
/*   connected to this group.                                    */ 11080000
/*                                                               */ 11090000
/*   The default is TTYGROUP(TTY).                               */ 11100000
/*                                                               */ 11110000
/*****************************************************************/ 11120000
  TTYGROUP(TTY)                                                    11130000
                                                                   11140000
```

SYS1.SAMPLIB(BPXISEC1)

```
/*************************************************************/
/*  Block 8 - Defining the UUCP user ID and UUCPG group ID   */
/*                                                           */
/*  If you plan on using the uucp utility and you use uppercase */
/*  group and user IDs on your system these will need to be setup. */
/*                                                           */
/* Customizable entries:                                     */
/* ====================                                      */
/*  a) yyyy - A unique GID value.                            */
/*  b) zzz - A unique UID value.                             */
/*  c) xxxxxxx - The password to be used.                    */
/*                                                           */
/*************************************************************/

/* ADDGROUP UUCPG OMVS(GID(yyyy))                           */

/* ADDUSER UUCP DFLTGRP(UUCPG) PASSWORD(xxxxxxx) +          */
/* OMVS(UID(zzz) HOME('/usr/spool/uucppublic') PROGRAM('/bin/sh')) */
```

SYS1.SAMPLIB(BPXISEC1)

```
/*************************************************************/
/*  Block 9 - Enable automatic assignment of unique UNIX     */
/*            identities                                     */
/* To allow users access to z/OS UNIX services without adding*/
/* a unique OMVS segment to each user ID.  Users without OMVS*/
/* segment data will be assigned a unique UID value the      */
/* first time they access a z/OS UNIX function or resource.  */
/*                                                           */
/* Customizable entries:                                     */
/* ====================                                      */
/*  APPLDATA on the BPX.NEXT.USER facility                   */
/*   specify the starting value (or a range of values)       */
/*   for assigning UID and GID values.                       */
/*    APPLDATA('xxxx/yyyy')                                  */
/*     xxxx is the starting value for assigning UID values   */
/*     yyyy is the starting value for assigning GID values   */
/*   Alternately, the UID and GID values can be a range of values */
/*    APPLDATA('xxxx-nnnn/yyyy-mmmm')                        */
/*                                                           */
/*************************************************************/

/* RDEFINE UNIXPRIV SHARED.IDS UACC(NONE)                    */
/* SETROPTS CLASSACT(UNIXPRIV) RACLIST(UNIXPRIV)             */
/* RDEFINE FACILITY BPX.NEXT.USER APPLDATA('2001/201')       */
/* RDEFINE FACILITY BPX.UNIQUE.USER                          */

/* SETROPTS RACLIST(FACILITY) REFRESH                        */
```

SYS1.SAMPLIB(BPXISEC1)

```
/*************************************************************/
/*  Block 10 - Define a model user profile defining OMVS segment    */
/*             information.  When a user is assigned a UNIQUE UID,  */
/*             they will also be assigned OMVS information from     */
/*             this model.                                          */
/* Continued from Block 9.                                         */
/*                                                                 */
/* Customizable entries:                                           */
/* ====================                                            */
/*  Specifying a HOME directory of /u/&racuid will assign          */
/*  a home directory of /u/userid to the user.                     */
/*  (If userid USERABC has an OMVS segemnt created as a result     */
/*   of BPX.UNIQUE.USER processing, the home directory             */
/*   will be /u/userabc.)                                          */
/*                                                                 */
/* If the BPX.UNIQUE.USER is already defined, replace RDEFINE with */
/* RALTER when setting the APPLDATA                                */
/*************************************************************/

/* ADDUSER BPXMODEL NAME('OMVS model user profile')                */
/*     OMVS(HOME('/u/&racuid') PROGRAM('/bin/sh'))                 */
/*     NOPASSWORD RESTRICTED                                       */

/*  RDEFINE FACILITY BPX.UNIQUE.USER APPLDATA('BPXMODEL')          */
/*  SETROPTS RACLIST(FACILITY) REFRESH                             */

```

SYS1.SAMPLIB(BPXISEC1)

```
/*************************************************************/
/*  Block 11 - Defining Started Tasks to z/OS UNIX           */
/*                                                           */
/* Any started task, like RMFGAT, that requires the use of z/OS */
/* UNIX services, must be defined to the RACF STARTED class. */
/* Customers who want to use the started procedure table (ICHRIN03) */
/* instead, only need a UID and GID is assigned to its user ID and */
/* group ID. This is done with the ADDUSER command below.    */
/*                                                           */
/* Customers implementing the BPX.DEFAULT.USER profile, do not */
/* need to specify the OMVS UID on the ADDUSER command below. */
/*                                                           */
/* Customizable entries:                                     */
/* ====================                                      */
/*  a) RMFGAT - refers to the name of started task.          */
/*  b) If you decide the started task requires to be trusted, */
/*     change TRUSTED(NO) to TRUSTED(YES) below.             */
/*  c) zzz - A unique UID value.                             */
/*                                                           */
/*************************************************************/

/* SETROPTS GENERIC(STARTED)                                 */
/* RDEFINE STARTED RMFGAT.* STDATA(USER(RMFGAT) GROUP(OMVSGRP) + */
/*   TRUSTED(NO))                                            */
/* SETROPTS CLASSACT(STARTED) RACLIST(STARTED)               */
/* SETROPTS RACLIST(STARTED) REFRESH                         */
 /* ADDUSER RMFGAT OMVS(UID(zzz)) DFLTGRP(OMVSGRP)           */
```

SYS1.SAMPLIB(BPXISEC1)

```
/*****************************************************************/
/*  Block 12 - Colony Address Spaces                             */
/*                                                               */
/* If you are defining colony address spaces for a physical file */
/* system (for example, for the NFS Client).                     */
/* Customers who want to use the started procedure table (ICHRIN03) */
/* instead, should refer to the z/OS UNIX Planning book.         */
/*                                                               */
/* Customizable entries:                                         */
/* ====================                                          */
/*  a) yyyyyyyy - refers to the entry for ASNAME specified on the */
/*     FILESYSTYPE statement in the BPXPRMxx member.             */
/*                                                               */
/*****************************************************************/

/* SETROPTS GENERIC(STARTED)                                     */
/* RDEFINE STARTED yyyyyyyy.* STDATA(USER(OMVSKERN) GROUP(OMVSGRP) +*/
/*   TRUSTED(NO))                                                */
/* SETROPTS CLASSACT(STARTED) RACLIST(STARTED)                   */
/* SETROPTS RACLIST(STARTED) REFRESH                             */

```

SYS1.SAMPLIB(BPXISEC1)

```
/*****************************************************************/
/*  Block 13 - BPX.SUPERUSER FACILITY class                      */
/*                                                               */
/* To allow non-zero UID users who can switch to become superuser,  */
/* the BPX.SUPERUSER Facility class will need to be setup. Next,    */
/* those users will need to be permitted to this facility class.    */
/*                                                               */
/* Customizable entries:                                         */
/* =====================                                         */
/*  a) zzzzzzzz - refers to the non-zero UID user ID to be       */
/*     permitted to this FACILITY class.                         */
/*                                                               */
/*****************************************************************/

/* RDEFINE FACILITY BPX.SUPERUSER UACC(NONE)                     */
/* SETROPTS CLASSACT(FACILITY)                                   */
/* SETROPTS RACLIST(FACILITY)                                    */

/* PERMIT BPX.SUPERUSER CLASS(FACILITY) ID(zzzzzzzz) ACCESS(READ)   */
/* SETROPTS RACLIST(FACILITY) REFRESH                            */

```

SYS1.SAMPLIB(BPXISEC1)

```
/*********************************************************************/
/*  Block 14 - BPX.DAEMON FACILITY class                             */
/*                                                                   */
/* If BPX.DAEMON facility class is defined, your system can          */
/* exercise more control over your superusers and greater control    */
/* over z/OS resources that a user (or daemon) can access.           */
/*                                                                   */
/* Customizable entries:                                             */
/* ====================                                              */
/*  a) zzzzzzzz - refers to the daemon to be permitted to this       */
/*     FACILITY class.                                               */
/*                                                                   */
/*********************************************************************/

/* RDEFINE FACILITY BPX.DAEMON UACC(NONE)                            */
/* SETROPTS CLASSACT(FACILITY)                                       */
/* SETROPTS RACLIST(FACILITY)                                        */

/* PERMIT BPX.DAEMON CLASS(FACILITY) ID(zzzzzzzz) ACCESS(READ)       */
/* SETROPTS RACLIST(FACILITY) REFRESH                                */

```

SYS1.SAMPLIB(BPXISEC1)

```
/*****************************************************************/
/*  Block 15 - BPX.SERVER FACILITY class                         */
/*                                                               */
/* If BPX.SERVER facility class is defined, your system can      */
/* exercise more control over your superusers and greater control*/
/* over z/OS resources that a user (or daemon) can access at a   */
/* task or thread level.                                         */
/*                                                               */
/* Customizable entries:                                         */
/* ====================                                          */
/*  a) zzzzzzzz - refers to the server program (ie. WEBSERV) to be*/
/*     permitted to this FACILITY class.                         */
/*                                                               */
/*  b) UPDATE access allows the server to establish a thread level*/
/*     (task-level) security environment for clients connecting to*/
/*     the server. Decisions about access control for z/OS       */
/*     resources (such as data sets) and to z/OS UNIX resources  */
/*     (such as HFS files) that are accessed by the client's thread*/
/*     in the server are made using only the RACF identity of the*/
/*     client.                                                   */
/*                                                               */
/*  c) READ access allows the server to establish a thread-level */
/*     security environment for the clients that it services.    */
/*     However, unless the server application has specified a valid*/
/*     RACF password or PassTicket on the pthread_security_np    */
/*     service invocation, the user ID of the server and the     */
/*     user ID of the client are used in resource access contol  */
/*     decisions.                                                */
/*                                                               */
/*****************************************************************/

/* RDEFINE FACILITY BPX.SERVER UACC(NONE)                        */
/* SETROPTS CLASSACT(FACILITY)                                   */
/* SETROPTS RACLIST(FACILITY)                                    */

/* PERMIT BPX.SERVER CLASS(FACILITY) ID(zzzzzzzz) ACCESS(READ)   */
/* SETROPTS RACLIST(FACILITY) REFRESH                            */
```

SYS1.SAMPLIB(BPXISEC1)

```
/*************************************************************/
/*  Block 16 - BPX.SMF FACILITY class                        */
/*                                                           */
/* To restrict access for C/C++ applications to generate SMF */
/* records without requiring APF authorization.              */
/*                                                           */
/* Customizable entries:                                     */
/* =====================                                     */
/*  a) zzzzzzzz - refers to the non-zero UID user ID to be   */
/*     permitted to this FACILITY class.                     */
/*                                                           */
/*************************************************************/

/* RDEFINE FACILITY BPX.SMF UACC(NONE)                       */
/* SETROPTS CLASSACT(FACILITY)                               */
/* SETROPTS RACLIST(FACILITY)                                */

/* PERMIT BPX.SMF CLASS(FACILITY) ID(zzzzzzzz) ACCESS(READ)  */
/* SETROPTS RACLIST(FACILITY) REFRESH                        */

```

SYS1.SAMPLIB(BPXISEC1)

```
/*************************************************************/
/*  Block 17 - BPX.DEBUG FACILITY class                      */
/*                                                           */
/* To allow users to use dbx to debug programs that run APF  */
/* authorized or with BPX.SERVER authority.                  */
/*                                                           */
/* Customizable entries:                                     */
/* =====================                                     */
/*  a) zzzzzzzz - refers to the non-zero UID user ID to be   */
/*     permitted to this FACILITY class.                     */
/*                                                           */
/*************************************************************/

/* RDEFINE FACILITY BPX.DEBUG UACC(NONE)                     */
/* SETROPTS CLASSACT(FACILITY)                               */
/* SETROPTS RACLIST(FACILITY)                                */

/* PERMIT BPX.DEBUG CLASS(FACILITY) ID(zzzzzzzz) ACCESS(READ)*/
/* SETROPTS RACLIST(FACILITY) REFRESH                        */

```

SYS1.SAMPLIB(BPXISEC1)

```
/*************************************************************/
/*  Block 18 - BPX.WLMSERVER FACILITY class                  */
/*                                                           */
/* To allow users to use WLM server functions.               */
/*                                                           */
/* Customizable entries:                                     */
/* ====================                                      */
/*  a) zzzzzzzz - refers to the non-zero UID user ID to be   */
/*     permitted to this FACILITY class.                     */
/*                                                           */
/*************************************************************/

/* RDEFINE FACILITY BPX.WLMSERVER UACC(NONE)                 */
/* SETROPTS CLASSACT(FACILITY)                               */
/* SETROPTS RACLIST(FACILITY)                                */

/* PERMIT BPX.WLMSERVER CLASS(FACILITY) ID(zzzzzzzz) ACCESS(READ)   */
/* SETROPTS RACLIST(FACILITY) REFRESH                        */

```

SYS1.SAMPLIB(BPXISEC1)

```
/*************************************************************/
/*  Block 19 - BPX.STOR.SWAP FACILITY class                  */
/*                                                           */
/* To allow users to make address spaces nonswappable.       */
/*                                                           */
/* Customizable entries:                                     */
/* ====================                                      */
/*  a) zzzzzzzz - refers to the non-zero UID user ID to be   */
/*     permitted to this FACILITY class.                     */
/*                                                           */
/*************************************************************/

/* RDEFINE FACILITY BPX.STOR.SWAP UACC(NONE)                 */
/* SETROPTS CLASSACT(FACILITY)                               */
/* SETROPTS RACLIST(FACILITY)                                */

/* PERMIT BPX.STOR.SWAP CLASS(FACILITY) ID(zzzzzzzz) ACCESS(READ)   */
/* SETROPTS RACLIST(FACILITY) REFRESH                        */

```

SYS1.SAMPLIB(BPXISEC1)

```
/*************************************************************/
/*  Block 20 - BPX.FILEATTR.APF FACILITY class               */
/*                                                           */
/* To allow users to turn on the APF-authorized attribute for an    */
/* HFS file.                                                 */
/*                                                           */
/* Customizable entries:                                     */
/* =====================                                     */
/*  a) zzzzzzzz - refers to the user ID to be permitted      */
/*     to this FACILITY class.                               */
/*                                                           */
/*************************************************************/

/* RDEFINE FACILITY BPX.FILEATTR.APF UACC(NONE)              */
/* SETROPTS CLASSACT(FACILITY)                               */
/* SETROPTS RACLIST(FACILITY)                                */

/* PERMIT BPX.FILEATTR.APF CLASS(FACILITY) ID(zzzzzzzz) ACCESS(READ)*/
/* SETROPTS RACLIST(FACILITY) REFRESH                        */
```

SYS1.SAMPLIB(BPXISEC1)

```
/*************************************************************/
/*  Block 21 - BPX.FILEATTR.PROGCTL FACILITY class           */
/*                                                           */
/* To allow users to turn on the program-controlled attribute for   */
/* an HFS file.                                              */
/*                                                           */
/* Customizable entries:                                     */
/* =====================                                     */
/*  a) zzzzzzzz - refers to the user ID to be permitted      */
/*     to this FACILITY class.                               */
/*                                                           */
/*************************************************************/

/* RDEFINE FACILITY BPX.FILEATTR.PROGCTL UACC(NONE)          */
/* SETROPTS CLASSACT(FACILITY)                               */
/* SETROPTS RACLIST(FACILITY)                                */

/* PERMIT BPX.FILEATTR.PROGCTL CLASS(FACILITY) ID(zzzzzzzz) +       */
/*   ACCESS(READ)                                            */
/* SETROPTS RACLIST(FACILITY) REFRESH                        */

```

SYS1.SAMPLIB(BPXISEC1)

```
/*************************************************************/
/*  Block 22 - BPX.FILEATTR.SHARELIB FACILITY Class Profile  */
/*                                                           */
/* To require extra privilege when setting and unsetting the */
/* shared library extended attribute via the chattr callable */
/* service.                                                  */
/*                                                           */
/* Customizable entries:                                     */
/* ====================                                      */
/*  a) zzzzzzzz - refers to the user ID to be permitted      */
/*     to this FACILITY class profile.                       */
/*                                                           */
/*************************************************************/

/* RDEFINE FACILITY BPX.FILEATTR.SHARELIB UACC(NONE)         */
/* SETROPTS CLASSACT(FACILITY)                               */
/* SETROPTS RACLIST(FACILITY)                                */

/* PERMIT BPX.FILEATTR.SHARELIB CLASS(FACILITY) ID(zzzzzzzz)   +    */
/* ACCESS(READ)                                              */

/* SETROPTS RACLIST(FACILITY) REFRESH                        */

```

SYS1.SAMPLIB(BPXISEC1)

```
/*************************************************************/
/*  Block 23 - BPX.FILEATTR.* generic FACILITY class         */
/*                                                           */
/* Instead of creating discrete FACILITY class profiles for  */
/* BPX.FILEATTR.APF and BPX.FILEATTR.PROGCTL and the         */
/* BPX.FILEATTR.SHARELIB, a generic FACILITY profile called  */
/* profile called BPX.FILEATTR.* can be created, to allow users */
/* to turn on the program-controlled and APF-authorized attribute. */
/*                                                           */
/* Customizable entries:                                     */
/* ====================                                      */
/*  a) zzzzzzzz - refers to the user ID to be permitted      */
/*     to this generic FACILITY class.                       */
/*                                                           */
/*************************************************************/

/* SETROPTS GENERIC(FACILITY)                                */
/* RDEFINE FACILITY BPX.FILEATTR.* UACC(NONE)                */
/* SETROPTS CLASSACT(FACILITY)                               */
/* SETROPTS RACLIST(FACILITY)                                */

/* PERMIT BPX.FILEATTR.* CLASS(FACILITY) ID(zzzzzzzz) ACCESS(READ)  */
/* SETROPTS RACLIST(FACILITY) REFRESH                        */
```

SYS1.SAMPLIB(BPXISEC1)

```
/*************************************************************/
/*  Block 24 - Removing users from a FACILITY CLASS          */
/*                                                           */
/* If you no longer need to give authority to a particular   */
/* FACILITY class (ie. BPX.SUPERUSER) to a certain user.     */
/*                                                           */
/* Customizable entries:                                     */
/* =====================                                     */
/*  a) xxxxxxxx - refers to the particluar FACILITY class.   */
/*  b) zzzzzzzz - refers to the UID user ID which no longer  */
/*     requires access.                                      */
/*                                                           */
/*************************************************************/

/* PERMIT BPX.xxxxxxxx CLASS(FACILITY) ID(zzzzzzzz) DELETE   */
/* SETROPTS RACLIST(FACILITY) REFRESH                        */

```

SYS1.SAMPLIB(BPXISEC1)

```
/*****************************************************************/
/*  Block 25 - Setting Up field-level Access for the OMVS Segment */
/*                                                                */
/* To allow a user to see or change OMVS fields in a RACF user    */
/* profile.                                                       */
/*                                                                */
/* Customizable entries:                                          */
/* ====================                                           */
/*  a) &RACUID allows all users to look at their own fields.      */
/*     Note that for this clist to execute, two '&' are required. */
/*     If you chose to run this PERMIT command outside of a clist,*/
/*     use only 1 '&' with the &RACUID variable.                  */
/*  b) READ access allows users to read the UID field.            */
/*  c) UPDATE access allows users to update their home directory  */
/*     in the HOME field or the program invoked for a TSO/E OMVS  */
/*     command in the PROGRAM field.                              */
/*                                                                */
/*****************************************************************/

/* RDEFINE FIELD USER.OMVS.UID  UACC(NONE)                        */
/* RDEFINE FIELD USER.OMVS.HOME UACC(NONE)                        */
/* RDEFINE FIElD USER.OMVS.PROGRAM UACC(NONE)                     */

/* SETROPTS CLASSACT(FIELD)                                       */
/* SETROPTS RACLIST(FIELD)                                        */

/* PERMIT USER.OMVS.UID  CLASS(FIELD) ID(&&RACUID) ACCESS(READ)   */
/* PERMIT USER.OMVS.HOME CLASS(FIELD) ID(&&RACUID) ACCESS(UPDATE) */
/* PERMIT USER.OMVS.PROGRAM CLASS(FIELD) ID(&&RACUID) ACCESS(UPDATE)*/

/* SETROPTS RACLIST(FIELD) REFRESH                                */
```

SYS1.SAMPLIB(BPXISEC1)

```
/*************************************************************/
/*  Block 26 - Defining Servers to Process Users without Passwords  */
/*                                                           */
/* Customizable entries:                                     */
/* =====================                                     */
/*  a) xxxxxxxx - the name of the server.                    */
/*  b) ANONYMOS - the MVS user ID of the user that the server will  */
/*     act as a surrogate of.                                */
/*                                                           */
/*************************************************************/

/* SETROPTS CLASSACT(SURROGAT)                               */
/* SETROPTS RACLIST(SURROGAT)                                */

/* RDEFINE SURROGAT BPX.SRV.ANONYMOS UACC(NONE)              */
/* SETROPTS RACLIST(SURROGAT) REFRESH                        */

/* PERMIT BPX.SRV.ANONYMOS CLASS(SURROGAT) ID(xxxxxxxx) +    */
/*    ACCESS(READ)                                           */
/* SETROPTS RACLIST(SURROGAT) REFRESH                        */

```

SYS1.SAMPLIB(BPXISEC1)

```
/*****************************************************************/
/*  Block 27 - Defining data sets to Program Control             */
/*                                                               */
/* All programs loaded into an address space that requires daemon*/
/* authority need to be marked controlled. These set of          */
/* instructions will mark an entire data set as program controlled.*/
/*                                                               */
/* You will also need to make the LE runtime library (SCEERUN)   */
/* program controlled.                                           */
/*                                                               */
/* Customizable entries:                                         */
/* =====================                                         */
/*  a) If you already have the PROGRAM * profile defined, then   */
/*     enter the RALTER command instead of RDEFINE below.        */
/*  b) SYS1.LINKLIB - the name of data set you want program      */
/*     controlled.                                               */
/*  c) VOLSER - The name of the volume where data set resides.   */
/*     This can be substitued with '******' (including single    */
/*     quotes) to refer to the SYSRES. VOLSER can also be        */
/*     omitted, to mean any volume where the library resides.    */
/*                                                               */
/* Note: Defining * for SYS1.LINKLIB with UACC(READ) enables anyone */
/*       to run the RACF Data Security Monitor Program and the RACF */
/*       Dynamic Parse Initialization command. You may want to   */
/*       define discrete profiles for these with UACC of NONE.   */
/*       The profile names are ICHDSM00 and IRRDPI00. You can then */
/*       explicitly grant read access only to certain users.     */
/*                                                               */
/*****************************************************************/

/* SETROPTS WHEN(PROGRAM)                                        */

/* RDEFINE PROGRAM * ADDMEM('SYS1.LINKLIB'/VOLSER/NOPADCHK) +    */
/*   UACC(READ)                                                  */

/* RDEFINE PROGRAM ICHDSM00   +                                  */
/*   ADDMEM('SYS1.LINKLIB'/VOLSER/NOPADCHK) UACC(NONE)           */
/* RDEFINE PROGRAM IRRDPI00   +                                  */
/*   ADDMEM('SYS1.LINKLIB'/VOLSER/NOPADCHK) UACC(NONE)           */

/* SETROPTS WHEN(PROGRAM) REFRESH                                */
```

SYS1.SAMPLIB(BPXISEC1)

```
/*************************************************************/
/*  Block 28 - Defining specific programs to Program Control */
/*                                                           */
/* If you choose not to define the data set where daemon programs */
/* reside (ie. SYS1.LINKLIB) as program controlled, then you can  */
/* define specific daemon programs as program controlled.         */
/*                                                           */
/* Refer to the z/OS UNIX System Service Planning book for the    */
/* list of specific daemon programs.                         */
/*                                                           */
/* Customizable entries:                                     */
/* ====================                                      */
/*  a) xxxxxx - The name of the daemon program.              */
/*  b) SYS1.LINKLIB - the name of data set you want program  */
/*     controlled.                                           */
/*  c) VOLSER - The name of the volume where data set resides. */
/*     This can be substitued with '******' (including single */
/*     quotes) to refer to the SYSRES. VOLSER can also be     */
/*     omitted, to mean any volume where the library resides. */
/*                                                           */
/*************************************************************/

/* SETROPTS WHEN(PROGRAM)                                    */
/* RDEFINE PROGRAM xxxxxx ADDMEM('SYS1.LINKLIB'/VOLSER/NOPADCHK) + */
/*   UACC(READ)                                              */
/* SETROPTS WHEN(PROGRAM) REFRESH                            */

```

SYS1.SAMPLIB(BPXISEC1)

```
/*************************************************************/
/*  Block 29 - Removing data sets from Program Control       */
/*                                                           */
/* To remove protection of libraries from program control.   */
/*                                                           */
/* Customizable entries:                                     */
/* ====================                                      */
/*  a) SYS1.LINKLIB - the name of data set you want program  */
/*     controlled.                                           */
/*  b) VOLSER - The name of the volume where data set resides.*/
/*     This can be substitued with '******' (including single*/
/*     quotes) to refer to the SYSRES. VOLSER can also be    */
/*     omitted, to mean any volume where the library resides.*/
/*                                                           */
/*************************************************************/

/* RALTER PROGRAM * DELMEM ('SYS1.LINKLIB'/VOLSER/NOPADCHK)  */
/* SETROPTS WHEN(PROGRAM) REFRESH                            */

```

SYS1.SAMPLIB(BPXISEC1)

```
/*************************************************************/
/*  Block 30 - Removing specific programs from Program Control */
/*                                                             */
/* To remove protection of specific programs from program control. */
/*                                                             */
/* Customizable entries:                                       */
/* ====================                                        */
/*  a) xxxxxx - The name of the daemon program.                */
/*  b) SYS1.LINKLIB - the name of data set you want program    */
/*     controlled.                                             */
/*  c) VOLSER - The name of the volume where data set resides. */
/*     This can be substitued with '******' (including single  */
/*     quotes) to refer to the SYSRES. VOLSER can also be      */
/*     omitted, to mean any volume where the library resides.  */
/*                                                             */
/*************************************************************/

/* RALTER PROGRAM xxxxxx DELMEM ('SYS1.LINKLIB'/VOLSER/NOPADCHK)  */
/* SETROPTS WHEN(PROGRAM) REFRESH                              */

```

SYS1.SAMPLIB(BPXISEC1)

```
/*************************************************************/
/*  Block 31 - Removing entire profiles from Program Control */
/*                                                           */
/* To remove the entire profile if there was only one library */
/* for a specific program.                                   */
/*                                                           */
/* Customizable entries:                                     */
/* =====================                                     */
/*  a) xxxxxx - The name of the daemon program.              */
/*                                                           */
/*************************************************************/

/* RDELETE PROGRAM xxxxxx                                    */
/* SETROPTS WHEN(PROGRAM) REFRESH                            */

```

SYS1.SAMPLIB(BPXISEC1)

```
/*************************************************************/
/*  Block 32 - Refreshing RACF In-Storage Data               */
/*                                                           */
/*************************************************************/

/* SETROPTS WHEN(PROGRAM) REFRESH                            */
/* SETROPTS RACLIST(FACILITY) REFRESH                        */
/* SETROPTS RACLIST(SURROGAT) REFRESH                        */
/* SETROPTS RACLIST(STARTED) REFRESH                         */
/* SETROPTS RACLIST(FIELD) REFRESH                           */
/* SETROPTS RACLIST(UNIXPRIV) REFRESH                        */

```

SYS1.SAMPLIB(BPXISEC1)

```
/*****************************************************************/
/*  Block 33 - BPX.SAFFASTPATH Profile in the FACILITY Class     */
/*                                                               */
/* When the BPX.SAFFASTPATH FACILITY Class profile is defined,   */
/* successful security checks are not audited.                   */
/*                                                               */
/* If this FACILITY class profile is defined before the system   */
/* is IPLed, the SAF fastpath support is enabled.                */
/*                                                               */
/* If defined after the system is IPLed, you must issue any of the */
/* SETOMVS or SET OMVS operator commands to activate this support. */
/*                                                               */
/* You can also start the refresh by issuing the following       */
/* command, where xx represents a BPXPRMxx member that is empty: */
/*    SET OMVS=(xx)                                              */
/*                                                               */
/* Users do not need to be permitted to this profile.            */
/*                                                               */
/* If your installation uses the IRRSXT00 exit to control        */
/* HFS access, do not define this profile.                       */
/*                                                               */
/*****************************************************************/

/* SETROPTS CLASSACT(FACILITY)                                   */
/* SETROPTS RACLIST(FACILITY)                                    */

/* RDEFINE FACILITY BPX.SAFFASTPATH UACC(NONE)                   */

/* SETROPTS RACLIST(FACILITY) REFRESH                            */
```

SYS1.SAMPLIB(BPXISEC1)

```
/*****************************************************************/
/*  Block 34 - Managing z/OS UNIX Privileges                     */
/*                                                               */
/*  By defining profiles in the UNIXPRIV Class, you can          */
/*  specifically grant certain UNIX superuser privileges to users*/
/*  who do not have UNIX superuser authority.                    */
/*                                                               */
/*  By minimizing the assignment of UNIX superuser authority, you*/
/*  reduce your security risk.                                   */
/*                                                               */
/*  Resource names in the UNIXPRIV Class are associated with     */
/*  z/OS UNIX privileges.                                        */
/*                                                               */
/*****************************************************************/
/*  Block 34.01 - SUPERUSER.FILESYS Profile in the               */
/*                UNIXPRIV Class                                 */
/*                                                               */
/*  Users granted READ access to this profile can read any       */
/*  HFS file, and read or search any HFS directory.              */
/*                                                               */
/*  Users granted UPDATE access to this profile can write to     */
/*  any HFS file, and includes privileges of READ access.        */
/*                                                               */
/*  Users granted CONTROL access (or higher) to this profile can */
/*  write to any HFS directory, and includes privileges of UPDATE*/
/*  access.                                                      */
/*                                                               */
/*  Note that there is no authorization to access Network File   */
/*  System (NFS) files provided by access to this profile.       */
/*                                                               */
/* Customizable entries:                                         */
/* =====================                                         */
/*  a) zzzzzzzz - refers to the non-zero UID user ID to be       */
/*     permitted to this privilege profile.                      */
/*  b) Change ACCESS(READ) to the specific level of authority    */
/*     that is to be granted.                                    */
/*****************************************************************/

/* SETROPTS CLASSACT(UNIXPRIV)                                   */
/* SETROPTS RACLIST(UNIXPRIV)                                    */

/* RDEFINE UNIXPRIV SUPERUSER.FILESYS UACC(NONE)                 */

/* PERMIT SUPERUSER.FILESYS CLASS(UNIXPRIV) ID(zzzzzzzz)  +      */
/* ACCESS(READ)                                                  */

/* SETROPTS RACLIST(UNIXPRIV) REFRESH                            */
```

SYS1.SAMPLIB(BPXISEC1)

```
/*************************************************************/
/*  Block 34.02 - SUPERUSER.FILESYS.CHOWN Profile in the     */
/*                UNIXPRIV Class                             */
/*                                                           */
/*  Users granted READ access to this profile can use the 'chown'   */
/*  shell command or the ISPF shell environment to change the        */
/*  ownership of any file.                                   */
/*                                                           */
/* Customizable entries:                                     */
/* ====================                                      */
/*  a) zzzzzzzz - refers to the non-zero UID user ID to be   */
/*     permitted to this privilege profile.                  */
/*************************************************************/

/* SETROPTS CLASSACT(UNIXPRIV)                               */
/* SETROPTS RACLIST(UNIXPRIV)                                */

/* RDEFINE UNIXPRIV SUPERUSER.FILESYS.CHOWN UACC(NONE)       */

/* PERMIT SUPERUSER.FILESYS.CHOWN CLASS(UNIXPRIV) ID(zzzzzzzz) +    */
/* ACCESS(READ)                                              */

/* SETROPTS RACLIST(UNIXPRIV) REFRESH                        */

```

SYS1.SAMPLIB(BPXISEC1)

```
/*****************************************************************/
/*  Block 34.03 - SUPERUSER.FILESYS.MOUNT Profile in the         */
/*                UNIXPRIV Class                                 */
/*                                                               */
/*  Users granted READ access to this profile can issue the      */
/*  mount command with the nosetuid option and unmount a         */
/*  file system mounted with the nosetuid option.                */
/*  Users can also change the attributes of the file system      */
/*  mounted with the nosetuid option with the 'chmount' shell    */
/*  command.                                                     */
/*                                                               */
/*  Users granted UPDATE access to this profile can issue the    */
/*  mount command with the setuid option and unmount a           */
/*  file system mounted with the setuid option.                  */
/*  Users can also change the attributes of the file system      */
/*  mounted with the setuid option with the 'chmount' shell      */
/*  command.                                                     */
/*                                                               */
/* Customizable entries:                                         */
/* ====================                                          */
/*  a) zzzzzzzz - refers to the non-zero UID user ID to be       */
/*     permitted to this privilege profile.                      */
/*  b) Change ACCESS(READ) to the specific level of authority    */
/*     that is to be granted.                                    */
/*****************************************************************/

/* SETROPTS CLASSACT(UNIXPRIV)                                   */
/* SETROPTS RACLIST(UNIXPRIV)                                    */

/* RDEFINE UNIXPRIV SUPERUSER.FILESYS.MOUNT UACC(NONE)           */

/* PERMIT SUPERUSER.FILESYS.MOUNT CLASS(UNIXPRIV) ID(zzzzzzzz)  +   */
/* ACCESS(READ)                                                  */

/* SETROPTS RACLIST(UNIXPRIV) REFRESH                            */
```

SYS1.SAMPLIB(BPXISEC1)

```
/*************************************************************/
/*  Block 34.04 - SUPERUSER.FILESYS.QUIESCE Profile in the   */
/*                UNIXPRIV Class                             */
/*                                                           */
/*  Users granted READ access to this profile can issue      */
/*  quiesce and unquiesce commands for a file system mounted */
/*  with the nosetuid option.                                */
/*                                                           */
/*  Users granted UPDATE access to this profile can issue    */
/*  quiesce and unquiesce commands for a file system mounted */
/*  with the setuid option.                                  */
/*                                                           */
/* Customizable entries:                                     */
/* =====================                                     */
/*  a) zzzzzzzz - refers to the non-zero UID user ID to be   */
/*     permitted to this privilege profile.                  */
/*  b) Change ACCESS(READ) to the specific level of authority*/
/*     that is to be granted.                                */
/*************************************************************/

/* SETROPTS CLASSACT(UNIXPRIV)                               */
/* SETROPTS RACLIST(UNIXPRIV)                                */

/* RDEFINE UNIXPRIV SUPERUSER.FILESYS.QUIESCE UACC(NONE)     */

/* PERMIT SUPERUSER.FILESYS.QUIESCE CLASS(UNIXPRIV) ID(zzzzzzzz) +  */
/* ACCESS(READ)                                              */

/* SETROPTS RACLIST(UNIXPRIV) REFRESH                        */

```

SYS1.SAMPLIB(BPXISEC1)

```
/*************************************************************/
/*  Block 34.05 - SUPERUSER.FILESYS.PFSCTL Profile in the    */
/*                UNIXPRIV Class                             */
/*                                                           */
/*  Users granted READ access to this profile can use the    */
/*  Physical File System callable service.  In C/C++, you would */
/*  use the pfsctl() function and in Assembler, the BPX1PCT  */
/*  callable service.                                        */
/*                                                           */
/* Customizable entries:                                     */
/* =====================                                     */
/*  a) zzzzzzzz - refers to the non-zero UID user ID to be   */
/*     permitted to this privilege profile.                  */
/*************************************************************/

/* SETROPTS CLASSACT(UNIXPRIV)                               */
/* SETROPTS RACLIST(UNIXPRIV)                                */

/* RDEFINE UNIXPRIV SUPERUSER.FILESYS.PFSCTL UACC(NONE)      */

/* PERMIT SUPERUSER.FILESYS.PFSCTL CLASS(UNIXPRIV) ID(zzzzzzzz) +  */
/* ACCESS(READ)                                              */

/* SETROPTS RACLIST(UNIXPRIV) REFRESH                        */
```

SYS1.SAMPLIB(BPXISEC1)

```
/*****************************************************************/
/*  Block 34.06 - SUPERUSER.FILESYS.VREGISTER Profile in the     */
/*                UNIXPRIV Class                                 */
/*                                                               */
/*  Servers granted READ access to this profile can register     */
/*  as a VFS server in z/OS UNIX.                                */
/*  In C/C++, you would use the v_reg() callable service, and    */
/*  in Assembler, you would use the BPX1VRG callable service.    */
/*                                                               */
/*  After a server is registered, appropriate privileges are not */
/*  needed for subsequent VFS functions.                         */
/*                                                               */
/*  Only servers, such as NFS servers, are authorized through    */
/*  this resource. Users who connect as clients through file     */
/*  servers systems, such as NFS, are not authorized through     */
/*  this resource.                                               */
/*                                                               */
/*  Refer to the z/OS UNIX System Services File System           */
/*  Interface book for more information on VFS servers.          */
/*                                                               */
/* Customizable entries:                                         */
/* ====================                                          */
/*  a) zzzzzzzz - refers to the server program (ie. WEBSERV) to be */
/*     permitted to this profile.                                */
/*****************************************************************/

/* SETROPTS CLASSACT(UNIXPRIV)                                   */
/* SETROPTS RACLIST(UNIXPRIV)                                    */

/* RDEFINE UNIXPRIV SUPERUSER.FILESYS.VREGISTER UACC(NONE)       */

/* PERMIT SUPERUSER.FILESYS.VREGISTER CLASS(UNIXPRIV)       +    */
/* ID(zzzzzzzz) ACCESS(READ)                                     */

/* SETROPTS RACLIST(UNIXPRIV) REFRESH                            */
```

SYS1.SAMPLIB(BPXISEC1)

```
/*************************************************************/
/*  Block 34.07 - SUPERUSER.IPC.RMID Profile in the UNIXPRIV Class  */
/*                                                           */
/*  Users granted READ access to this profile can issue the          */
/*  'ipcrm' shell command to release IPC resources such as           */
/*  message queues, semaphores set, or shared memory identifiers.    */
/*                                                           */
/* Customizable entries:                                     */
/* =====================                                     */
/*  a) zzzzzzzz - refers to the non-zero UID user ID to be           */
/*     permitted to this privilege profile.                  */
/*************************************************************/

/* SETROPTS CLASSACT(UNIXPRIV)                               */
/* SETROPTS RACLIST(UNIXPRIV)                                */

/* RDEFINE UNIXPRIV SUPERUSER.IPC.RMID UACC(NONE)            */

/* PERMIT SUPERUSER.IPC.RMID CLASS(UNIXPRIV)           +     */
/* ID(zzzzzzzz) ACCESS(READ)                                 */

/* SETROPTS RACLIST(UNIXPRIV) REFRESH                        */

```

SYS1.SAMPLIB(BPXISEC1)

```
/*************************************************************/
/*  Block 34.08 - SUPERUSER.PROCESS.GETPSENT Profile in the  */
/*                UNIXPRIV Class                             */
/*                                                           */
/*  Users granted READ access to this profile can use the    */
/*  w_getpsent callable service to receive data such as      */
/*  running times, user IDs (UIDs), group IDs (GIDs), and    */
/*  invocation parameters.                                   */
/*  In C/C++ you would use the w_getpsent function, and      */
/*  in Assembler, you would use the BPX1GPS callable service.*/
/*                                                           */
/* Customizable entries:                                     */
/* ====================                                      */
/*  a) zzzzzzzz - refers to the non-zero UID user ID to be   */
/*     permitted to this privilege profile.                  */
/*************************************************************/

/* SETROPTS CLASSACT(UNIXPRIV)                               */
/* SETROPTS RACLIST(UNIXPRIV)                                */

/* RDEFINE UNIXPRIV SUPERUSER.PROCESS.GETPSENT UACC(NONE)    */

/* PERMIT SUPERUSER.PROCESS.GETPSENT CLASS(UNIXPRIV) +       */
/* ID(zzzzzzzz) ACCESS(READ)                                 */

/* SETROPTS RACLIST(UNIXPRIV) REFRESH                        */
```

SYS1.SAMPLIB(BPXISEC1)

```
/*************************************************************/
/*  Block 34.09 - SUPERUSER.PROCESS.KILL Profile in the      */
/*                UNIXPRIV Class                             */
/*                                                           */
/*  Users granted READ access to this profile can use the    */
/*  kill callable service to send signals to any process.    */
/*  In C/C++, you would use the kill() function,              /
/*  and in Assembler, you would use the BPX1KIL callable service.  */
/*                                                           */
/* Customizable entries:                                     */
/* =====================                                     */
/*  a) zzzzzzzz - refers to the non-zero UID user ID to be   */
/*     permitted to this privilege profile.                  */
/*************************************************************/

/* SETROPTS CLASSACT(UNIXPRIV)                               */
/* SETROPTS RACLIST(UNIXPRIV)                                */

/* RDEFINE UNIXPRIV SUPERUSER.PROCESS.KILL UACC(NONE)        */

/* PERMIT SUPERUSER.PROCESS.KILL CLASS(UNIXPRIV) +           */
/* ID(zzzzzzzz) ACCESS(READ)                                 */

/* SETROPTS RACLIST(UNIXPRIV) REFRESH                        */

```

SYS1.SAMPLIB(BPXISEC1)

```
/*****************************************************************/
/*  Block 34.10 - SUPERUSER.PROCESS.PTRACE Profile in the        */
/*                UNIXPRIV Class                                 */
/*                                                               */
/*  Users granted READ access to this profile can use the        */
/*  ptrace function through the dbx debugger to trace any        */
/*  process, and are also allowed to use the 'ps' shell command  */
/*  to output information on all processes.                      */
/*  In C/C++, you would use the ptrace() function,               */
/*  and in Assembler you would use the BPX1PTR callable service. */
/*                                                               */
/*  Note that authorization to the resource BPX.DEBUG in the     */
/*  FACILITY class is also required to trace processes that run  */
/*  with APF authority or BPX.SERVER authority.                  */
/*                                                               */
/* Customizable entries:                                         */
/* =====================                                         */
/*  a) zzzzzzzz - refers to the non-zero UID user ID to be       */
/*     permitted to this privilege profile.                      */
/*****************************************************************/

/* SETROPTS CLASSACT(UNIXPRIV)                                   */
/* SETROPTS RACLIST(UNIXPRIV)                                    */

/* RDEFINE UNIXPRIV SUPERUSER.PROCESS.PTRACE UACC(NONE)          */

/* PERMIT SUPERUSER.PROCESS.PTRACE CLASS(UNIXPRIV) +             */
/* ID(zzzzzzzz) ACCESS(READ)                                     */

/* SETROPTS RACLIST(UNIXPRIV) REFRESH                            */
```

SYS1.SAMPLIB(BPXISEC1)

```
/*************************************************************/
/*  Block 34.11 - SUPERUSER.SETPRIORITY Profile in the       */
/*                UNIXPRIV Class                             */
/*                                                           */
/*  Users granted READ access to this profile can increase   */
/*  their own priority.                                      */
/*                                                           */
/* Customizable entries:                                     */
/* =====================                                     */
/*  a) zzzzzzzz - refers to the non-zero UID user ID to be   */
/*     permitted to this privilege profile.                  */
/*************************************************************/

/* SETROPTS CLASSACT(UNIXPRIV)                               */
/* SETROPTS RACLIST(UNIXPRIV)                                */

/* RDEFINE UNIXPRIV SUPERUSER.SETPRIORITY UACC(NONE)         */

/* PERMIT SUPERUSER.SETPRIORITY CLASS(UNIXPRIV) ID(zzzzzzzz)  +     */
/* ACCESS(READ)                                              */

/* SETROPTS RACLIST(UNIXPRIV) REFRESH                        */

```

SYS1.SAMPLIB(BPXISEC1)

```
/*************************************************************/
/*  Block 34.12 - CHOWN.UNRESTRICTED Profile in the UNIXPRIV Class  */
/*                                                           */
/*  Define and activate this profile to allow ALL z/OS UNIX          */
/*  users to transfer ownership of files they own to any UID or      */
/*  GID on the system.                                       */
/*                                                           */
/*************************************************************/

/* SETROPTS CLASSACT(UNIXPRIV)                               */
/* SETROPTS RACLIST(UNIXPRIV)                                */

/* RDEFINE UNIXPRIV CHOWN.UNRESTRICTED                       */

/* SETROPTS RACLIST(UNIXPRIV) REFRESH                        */

```

SYS1.SAMPLIB(BPXISEC1)

```
/*****************************************************************/
/*  Block 34.13 - Defining generic profiles for the UNIXPRIV Class  */
/*                                                                 */
/*  Generic profiles are allowed for resources in the UNIXPRIV     */
/*  class, with the exception of the CHOWN.UNRESTRICTED resource.  */
/*                                                                 */
/* Customizable entries:                                           */
/* =====================                                           */
/*  a) zzzzzzzz - refers to the non-zero UID user ID to be         */
/*     permitted to this privilege profile.                        */
/*  b) Change ACCESS(READ) to the specific level of authority     */
/*     that is to be granted.                                      */
/*****************************************************************/

/*****************************************************************/
/* Define generic profile for filesystem privileges.             */
/*****************************************************************/

/* SETROPTS GENERIC(UNIXPRIV)                                     */
/* SETROPTS CLASSACT(UNIXPRIV)                                    */
/* SETROPTS RACLIST(UNIXPRIV)                                     */

/* RDEFINE UNIXPRIV SUPERUSER.FILESYS.*  UACC(NONE)               */

/* PERMIT SUPERUSER.FILESYS.*  CLASS(UNIXPRIV) ID(zzzzzzzz)  +    */
/* ACCESS(READ)                                                   */

/* SETROPTS RACLIST(UNIXPRIV) REFRESH                             */
```

SYS1.SAMPLIB(BPXISEC1)

```
/*************************************************************/
/* Define generic profile for controling IPC resources.      */
/*************************************************************/

/* SETROPTS GENERIC(UNIXPRIV)                                */
/* SETROPTS CLASSACT(UNIXPRIV)                               */
/* SETROPTS RACLIST(UNIXPRIV)                                */

/* RDEFINE UNIXPRIV SUPERUSER.IPC.*  UACC(NONE)              */

/* PERMIT SUPERUSER.IPC.*  CLASS(UNIXPRIV) ID(zzzzzzzz)  +   */
/* ACCESS(READ)                                              */

/* SETROPTS RACLIST(UNIXPRIV) REFRESH                        */

```

SYS1.SAMPLIB(BPXISEC1)

```
/*************************************************************/
/* Define generic profile for process oriented resources.    */
/*************************************************************/

/* SETROPTS GENERIC(UNIXPRIV)                                */
/* SETROPTS CLASSACT(UNIXPRIV)                               */
/* SETROPTS RACLIST(UNIXPRIV)                                */

/* RDEFINE UNIXPRIV SUPERUSER.PROCESS.*  UACC(NONE)          */

/* PERMIT SUPERUSER.PROCESS.*  CLASS(UNIXPRIV) ID(zzzzzzzz)  +  */
/* ACCESS(READ)                                              */

/* SETROPTS RACLIST(UNIXPRIV) REFRESH                        */

```

SYS1.SAMPLIB(BPXISEC1)

```
/*************************************************************/
/*  Block 35 - BPX.JOBNAME FACILITY class profile            */
/*                                                           */
/* This profile controls which users are allowed to set their*/
/* own job names.  Non-superusers can define their own job names*/
/* on spawn or exec syscalls with READ or higher permission. */
/*                                                           */
/* Customizable entries:                                     */
/* ====================                                      */
/*  a) zzzzzzzz - refers to the user ID to be permitted      */
/*     to this FACILITY class.                               */
/*                                                           */
/*************************************************************/

/* RDEFINE FACILITY BPX.JOBNAME UACC(NONE)                   */
/* SETROPTS CLASSACT(FACILITY)                               */
/* SETROPTS RACLIST(FACILITY)                                */

/* PERMIT BPX.JOBNAME CLASS(FACILITY) ID(zzzzzzzz) ACCESS(READ)*/
/* SETROPTS RACLIST(FACILITY) REFRESH                        */

```

SYS1.SAMPLIB(BPXISEC1)

```
/*************************************************************/
/*  Block 35 - BPX.JOBNAME FACILITY class profile            */
/*                                                           */
/* This profile controls which users are allowed to set their*/
/* own job names.  Non-superusers can define their own job names*/
/* on spawn or exec syscalls with READ or higher permission. */
/*                                                           */
/* Customizable entries:                                     */
/* ====================                                      */
/*  a) zzzzzzzz - refers to the user ID to be permitted      */
/*     to this FACILITY class.                               */
/*                                                           */
/*************************************************************/

/* RDEFINE FACILITY BPX.JOBNAME UACC(NONE)                   */
/* SETROPTS CLASSACT(FACILITY)                               */
/* SETROPTS RACLIST(FACILITY)                                */

/* PERMIT BPX.JOBNAME CLASS(FACILITY) ID(zzzzzzzz) ACCESS(READ)*/
/* SETROPTS RACLIST(FACILITY) REFRESH                        */

```

SYS1.SAMPLIB(BPXISEC1)

```
/*************************************************************/
/*  Block 36 - BPX.MAP FACILITY class profile                */
/*                                                           */
/*  READ access to this profile allows applications to use the */
/*  __map_init (BPX1MMI) and the __map_service (BPX1MMS)     */
/*  callable services to create, connect, and disconnect mapped */
/*  megabyte areas.                                          */
/*                                                           */
/* Customizable entries:                                     */
/* ====================                                      */
/*  a) zzzzzzzz - refers to the application to be permitted  */
/*     to this FACILITY class.                               */
/*                                                           */
/*************************************************************/

/* RDEFINE FACILITY BPX.MAP  UACC(NONE)                      */
/* SETROPTS CLASSACT(FACILITY)                               */
/* SETROPTS RACLIST(FACILITY)                                */

/* PERMIT BPX.MAP CLASS(FACILITY) ID(zzzzzzzz) ACCESS(READ)  */
/* SETROPTS RACLIST(FACILITY) REFRESH                        */

```

SYS1.SAMPLIB(BPXISEC1)

```
/*************************************************************/
/*  Block 37 - BPX.DAEMON.HFSCTL FACILITY class profile      */
/*                                                           */
/* When a user (daemon) is permitted to this profile, program*/
/* control will be enforced only on HFS program objects.     */
/* You must activate the BPX.DAEMON facility class profile first.*/
/*                                                           */
/* Customizable entries:                                     */
/* ====================                                      */
/*  a) zzzzzzzz - refers to the daemon to be permitted       */
/*     to this FACILITY class.                               */
/*                                                           */
/*************************************************************/

/* RDEFINE FACILITY BPX.DAEMON.HFSCTL UACC(NONE)             */
/* SETROPTS CLASSACT(FACILITY)                               */
/* SETROPTS RACLIST(FACILITY)                                */

/* PERMIT BPX.DAEMON.HFSCTL CLASS(FACILITY) ID(zzzzzzzz) +   */
/*   ACCESS(READ)                                            */
/* SETROPTS RACLIST(FACILITY) REFRESH                        */

```

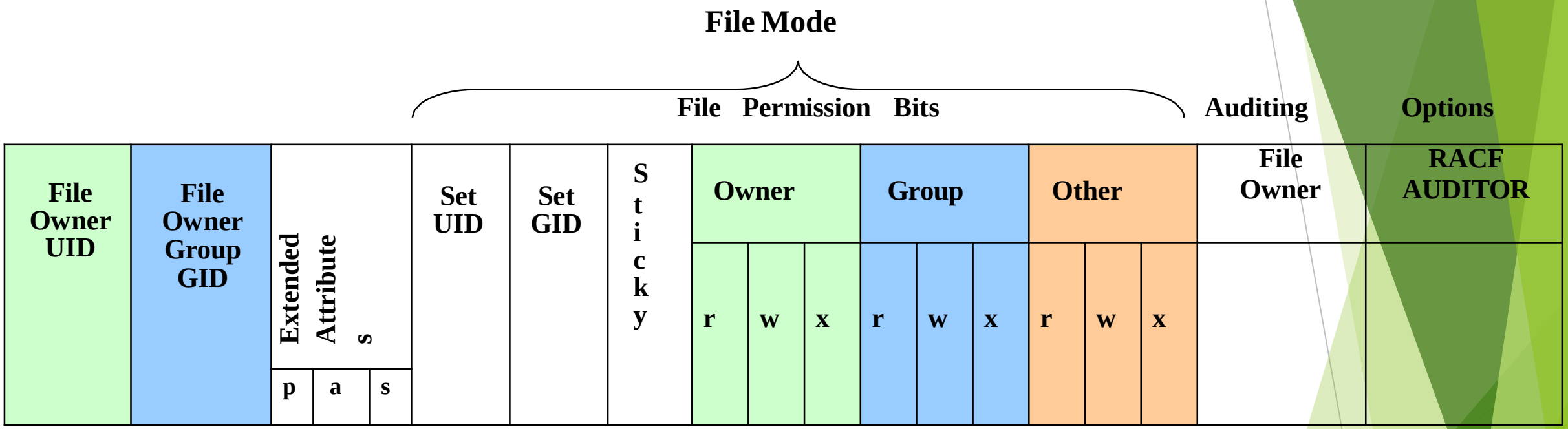
FACILITY Class

Resource Name	Authority Granted
BPX.CF	Controls the use of the Coupling Facility sizer tool (_cpl())
BPX.CONSOLE	Controls access to authorized features of the _console() service
BPX.DAEMON	Controls the change of MVS identities without knowing the target user ID's password
BPX.DAEMON.HFSCTL	Controls the loading of uncontrolled programs from MVS libraries into their address space
BPX.DEBUG	Controls the use of ptrace (via dbx) to debug prgrams
BPX.DEFAULT.USER	Defines the default uid and/or gid
BPX.FILEATTR.APF	Controls the setting of the APF-authorized attribute in an HFS file
BPX.FILEATTR.PROGCTL	Controls the setting of the program control attribute in an HFS file
BPX.FILEATTR.SHARELIB	Controls setting the shared library extended attribute in an HFS file
BPX.JOBNAME	Controls which users are allowed to set their own job names
BPX.MAINCHECK	Controls loading of MVS programs marked MAIN
BPX.MAP	Controls the use of storage mapping services
BPX.NEXT.USER	Enables automatic assignment of UIDs and GIDs.
BPX.OUTPUT.UNLIMITED	Allows users to override the default spooled output limits for processes
BPX.POE	Controls the use of Port-of-Entry for MLS security checks (_poe)
BPX.SAFFASTPATH	Enables faster security checks for file system
BPX.SERVER	Restricts the use of the pthread_security_np() service
BPX.SHUTDOWN	Controls special treatment at shutdown
BPX.SMF	Authorizes the cutting of SMF records
BPX.STOR.SWAP	Controls which users can make address spaces nonswappable
BPX.SUPERUSER	Allows users to switch to superuser authority
BPX.WLMSERVER	Controls access to the WLM server functions

The UNIXPRIV Class Profiles

Resource Name	Access Given
SUPERUSER.FILESYS (READ access)	Allows a user to read any HFS file and read or search any HFS directory.
SUPERUSER.FILESYS (UPDATE access)	Allows a user to write to any existing HFS file.
SUPERUSER.FILESYS (CONTROL access)	Allows a user to write to any HFS directory.
SUPERUSER.FILESYS.ACLOVERRIDE	Specifies that ACL entries override SUPERUSER.FILESYS
SUPERUSER.FILESYS.CHANGEPERMS	Allows users to change permission bits for any file.
SUPERUSER.FILESYS.CHOWN	Allows a user to change ownership of any file.
SUPERUSER.FILESYS.MOUNT	Allows a user to issue mount and unmount requests.
SUPERUSER.FILESYS.QUIESCE	Allows user to issue quiesce and unquiesce commands for a file system
SUPERUSER.FILESYS.PFSCTL	Allows a user to call pfsctl().
SUPERUSER.FILESYS.VREGISTER	Allows a user to issue vregister() to register as a vfs file server.
SUPERUSER.IPC.RMID	Allows a user to do ipcrm calls to clean up leftover IPC mechanisms.
SUPERUSER.PROCESS.GETPSENT	Allows user to see all processes.
SUPERUSER.PROCESS.KILL	Allows user to send signals to any process.
SUPERUSER.PROCESS.PTRACE	Allows user to use dbx to trace any process.
SUPERUSER.SETPRIORITY	Allows a user to increase his priority.

z/OS UNIX File Security Packet



File Permission Examples

-rwxrwxrwx = 777

A file anyone can read, write, execute

-rw-r--r-- = 644

A file the owner can read, write & anyone else can read

-rwx--x--- = 710

A file the owner can read, write, execute & group can execute

-rwxrw-rw- = 766

A file the owner can read, write, execute & anyone else can read, write (Update & Delete)

-rwxr-xr-x = 755

A file the owner can read, write, execute & anyone can read, execute

Displaying File Permissions

Output of the ls command

```
BOBS:/Z18/usr: >ls -la
total 304
drwxr-xr-x   12 BPXOINIT SYS1     8192 May 19   2006 .
drwxr-xr-x   11 BPXOINIT SYS1     8192 Oct 17   2006 ..
drwxr-xr-x    2 BPXOINIT SYS1     8192 May 19   2006 bin
drwxr-xr-x   11 BPXOINIT SYS1    45056 Oct  2   2006 include
drwxr-xr-x   10 BPXOINIT SYS1    28672 Oct 16   2006 lib
drwxr-xr-x    2 BPXOINIT SYS1     8192 Jan 31   2001 local
drwxr-xr-x   93 BPXOINIT SYS1     8192 Oct  2   2006 lpp
drwxrwxr-x    2 BPXOINIT SYS1     8192 Sep 17   1997 mail
drwxr-xr-x    5 BPXOINIT SYS1     8192 Jan 27   1999 man
drwxr-xr-x    3 BPXOINIT SYS1     8192 Oct 16   2006 sbin
drwxr-xr-x    3 BPXOINIT SYS1     8192 Jan 27   1999 share
drwxr-xr-x    6 BPXOINIT SYS1     8192 Sep 29   2004 spool
```

File Type

Permissions

Owning User

Owning Group

File Name

z/OS UNIX Basic Security

Overall z/OS UNIX Security design should pay careful attention to the implementation of user and group IDs, and group membership, and the permissions granted to these users and groups.

Be aware of the umask setting when creating new files!!

z/OS UID 0 Superuser Authority

- Passes all z/OS UNIX Security checks
- Perform administrative activities
- Install products
- Can access and modify any files and directories
- Not Limited to only z/OS UNIX component
- Manages all z/OS UNIX processes
- Can run unlimited number of processes concurrently
- Propagates superuser privileges to forked child process
- Can change identity

SYS1.PARMLIB(BPXPRMxx)

```
/*****************************************************************/ 01200000
/*                                                               */ 01210000
/*   Specify the maximum number of processes that z/OS UNIX      */ 01220000
/*   will allow to be active concurrently.                       */ 01230000
/*                                                               */ 01240000
/*   Notes:                                                      */ 01250000
/*                                                               */ 01260000
/*   1. Minimum allowable value is 5.                            */ 01270000
/*   2. Maximum allowable value is 32767.                        */ 01280000
/*   3. If this parameter is not provided, the system default    */ 01290000
/*      value for this parameter is 900.                         */ 01300000
/*                                                               */ 01310000
/*****************************************************************/ 01320000
  MAXPROCSYS(900)                 /* System will allow at most 900    01330000
                                     processes to be active           01340000
                                     concurrently             @P9C*/ 01350000
                                                                     01360000
```

SYS1.PARMLIB(BPXPRMxx)

```
/*****************************************************************/ 01370000
/*                                                               */ 01380000
/*   Specify the maximum number of processes that a single user  */ 01390000
/*   (that is, with the same UID) is allowed to have concurrently*/ 01400000
/*   active regardless of origin.                                */ 01410000
/*                                                               */ 01420000
/*   Notes:                                                      */ 01430000
/*                                                               */ 01440000
/*   1. This parameter is the same as the Child_Max variable     */ 01450000
/*      defined in POSIX 1003.1.                                 */ 01460000
/*   2. Minimum allowable value is 3.                            */ 01470000
/*   3. Maximum allowable value is 32767.                        */ 01480000
/*   4. If this parameter is not provided, the system default    */ 01490000
/*      value for this parameter is 25.                          */ 01500000
/*                                                               */ 01510000
/*****************************************************************/ 01520000
  MAXPROCUSER(25)                 /* Allow each user (same UID) to  01530000
                                     have at most 25 concurrent     01540000
                                     processes active            */ 01550000
                                                                    01560000
```

- **Overview**
- BPXPRMxx
- BPXISEC1
- Next Steps

z/OS UNIX Basic Security

Julie Bergh

berghju@gmail.com