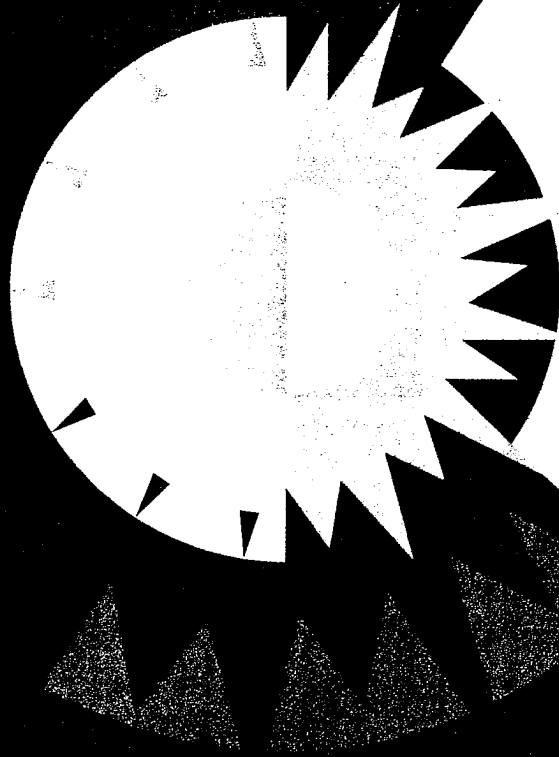
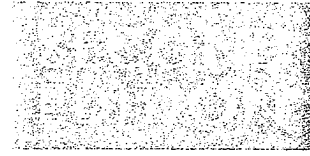


561

SOLARIS™

PORTING GUIDE



SunSoft Developer Engineering

 *SunSoft*



© 1995 Sun Microsystems, Inc. — Printed in the United States of America.
2550 Garcia Avenue, Mountain View, California 94043-1100 U.S.A.

All rights reserved. This book is protected by copyright and distributed under licenses restricting its use, copying, distribution, and decompilation. No part of this book may be reproduced in any form by any means without prior written authorization of Sun and its licensors, if any.

Portions of the products described in this book may be derived from the UNIX® and Berkeley 4.3 BSD systems, licensed from UNIX System Laboratories, Inc., a wholly owned subsidiary of Novell, Inc., and the University of California, respectively. Third-party font software in this product is protected by copyright and licensed from Sun's font suppliers.

RESTRICTED RIGHTS LEGEND: Use, duplication, or disclosure by the United States government is subject to restrictions as set forth in DFARS 252.227-7013 (c)(1)(ii) and FAR 52.227-19.

The products described in this book may be protected by one or more U.S. patents, foreign patents, or pending applications.

TRADEMARKS— Sun, Sun Microsystems, the Sun logo, SunSoft, Solaris, Solaris Sunburst Design, OpenWindows, ONC, ONC+, SunOS, DeskSet, ToolTalk, NFS, PC-NFS, NeWS, X11/NeWS, XView, OpenFonts, ProWorks/iMPact and SPARCworks/iMPact, XGL and XIL are trademarks or registered trademarks of Sun Microsystems, Inc. UNIX is a registered trademark in the United States and other countries exclusively licensed through X/Open Company, Ltd. OPEN LOOK® is a registered trademark of Novell, Inc. Adobe, PostScript, and Display PostScript are registered trademarks of Adobe Systems Incorporated. FrameMaker is a registered trademark of Frame Technology Corporation. Motif and OSF/Motif are registered trademarks of the Open Software Foundation. Pentium is a trademark and Intel is a registered trademark of Intel Corporation. X Window System is a trademark of X Consortium, Inc. Netscape is a trademark of Netscape Communications Corporation. All other product names mentioned herein are the trademarks of their respective owners.

All SPARC trademarks, including the SCD Compliant Logo, are trademarks or registered trademarks of SPARC International, Inc. SPARCstation, SPARCserver, SPARCengine, SPARCworks, and SPARCcompiler are licensed exclusively to Sun Microsystems, Inc. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

The OPEN LOOK® and Sun™ Graphical User Interfaces were developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees who implement OPEN LOOK GUIs and otherwise comply with Sun's written license agreements.

Code Examples 5-2 and 5-3 are taken from *The Programmer's Supplement for Release 5 of the X Window System, Version 11*, by David Flanagan (O'Reilly & Associates, Inc., 1991), and are used by permission. The Code Examples copyright © 1991 O'Reilly & Associates, Inc.

Material from X11 Release 5 documentation is reproduced with permission. X11 Release 5 documentation is copyright by the Massachusetts Institute of Technology, Cambridge, Massachusetts, and Digital Equipment Corporation, Maynard, Massachusetts.

The publisher offers discounts on this book when ordered in bulk quantities. For more information, contact: Corporate Sales Department, Prentice Hall PTR, One Lake Street, Upper Saddle River, NJ 07458, Phone: 800-382-3419 or 201-236-7156, Fax: 201-236-7141, email: corpsales@prenhall.com

Cover designer: *Anthony Gemmellaro*
Manufacturing manager: Alexis R. Heydt
Acquisitions editor: *Gregory G. Doench*
Editorial production and supervision: *Camille Trentacoste*

0 9 8 7 6 5 4 3 2 1

ISBN 0-13-443672-5

SunSoft Press
A Prentice Hall Title

Tab

Foreword

Preface . . .

Acknowledgments

Part 1: Introduction

Introduction

Why

Solaris

The

1. Standalone

Introduction

Why

Types

Source

The

Am

Acc

The

Stand

Maj

Gov

Solaris

UN.

X W

X/C

X/C

Table 15-4 lists the valid values for *cmd*, their meanings and the type of *arg*:

Table 15-4 Valid *prionctl.h* *cmd* Values

<i>cmd</i>	<i>arg</i> Type	Function
PC_GETCID	pcinfo_t	get class ID and attributes
PC_GETCLINFO	pcinfo_t	get class name and attributes
PC_SETPARMS	pcparms_t	set class and scheduling parameters
PC_GETPARMS	pcparms_t	get class and scheduling parameters

On success, *prionctl*(2) returns the following values:

- PC_GETCID returns the number of configured classes.
- PC_GETCLINFO returns the number of configured classes.
- PC_SETPARMS returns 0.
- PC_GETPARMS returns the PID of the process whose scheduler properties it is returning.

On failure, *prionctl*(2) returns -1 and sets *errno* to indicate the reason for the failure. See the man page for *prionctl*(2) for the complete list of error conditions.

The PC_GETCID and PC_GETCLINFO Commands

The PC_GETCID and PC_GETCLINFO commands retrieve scheduler parameters for a particular class based on the class ID or class name. Both commands use the following *pcinfo* structure to send arguments and receive return values:

```
typedef struct pcinfo {
    id_t pc_cid;           /* class id */
    char pc_clname[PC_CLNMSZ]; /* class name */
    long pc_clinfo[PC_CLINFOSZ]; /* class information */
} pcinfo_t;
```

The PC_GETCID command, given the class name, retrieves scheduler class ID and associated parameters. Some of the other *prionctl*(2) commands use the class ID to specify a scheduler class. The valid class names are TS for time-sharing and RT for realtime.

For the RT class, *pc_clinfo* contains an *rtinfo* structure which holds *rt_maxpri*, the maximum valid realtime priority; this value is configurable. In the default configuration, this is the highest priority any process can have. The minimum valid realtime priority is zero.

```
typedef struct rtinfo {
    short rt_maxpri;      /* maximum realtime priority */
} rtinfo_t;
```

For the TS class maximum time sharing end-use

```
typedef struct {
    short ts;
} tsinfo_t;
```

The following p RT scheduler cl

Code Example 15-1

```
/*
 * Get sched
 */

#include <sys/
#include <sys/
#include <sys/
#include <st
#include <st
#include <st
#include <er

main ()
{
    pcinfo_t
    tsinfo_t
    rtinfo_t
    short

    /* times
    (voic
    if (r
    F
    e
    )
    tsin:
    maxt:
    (voic
    F

    /* reall
    (voic
```

For the TS class, `pc_clinfo` contains a `tsinfo` structure which holds `ts_maxupri`, the maximum time-sharing end-user priority; this value is also configurable. The minimum time-sharing end-user priority is negative `ts_maxupri`.

```
typedef struct tsinfo {
    short ts_maxupri; /* limits of user priority range */
} tsinfo_t;
```

The following program, `getcid.c`, gets and prints the range of valid priorities for the TS and RT scheduler classes.

Code Example 15-1 getcid.c Program

```
/*
 * Get scheduler class IDs and priority ranges.
 */

#include <sys/types.h>
#include <sys/priocntl.h>
#include <sys/rtpriocntl.h>
#include <sys/tspriocntl.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <errno.h>

main ()
{
    pcinfo_t    pcinfo;
    tsinfo_t    *tsinfop;
    rtinfo_t*   rtinfop;
    short       maxtsupri, maxrtpri;

    /* timesharing */
    (void) strcpy (pcinfo.pc_clname, "TS");
    if (priocntl (0L, 0L, PC_GETCID, &pcinfo) == -1L) {
        perror ("PC_GETCID failed for timesharing class");
        exit (1);
    }
    tsinfop = (struct tsinfo *) pcinfo.pc_clinfo;
    maxtsupri = tsinfop->ts_maxupri;
    (void) printf("Timesharing: ID %ld, priority range -%d through %d\n",
        pcinfo.pc_cid, maxtsupri, maxtsupri);

    /* realtime */
    (void) strcpy(pcinfo.pc_clname, "RT");
```

Code Example 15-3 Program to Get Class Name Given Process ID (Continued)

```

        exit (3);
    }
    (void) printf("process ID %d, class %s\n", pid,
        pcinfo.pc_clname);
}

```

The PC_GETPARMS and PC_SETPARMS Commands

The PC_GETPARMS command retrieves and the PC_SETPARMS command sets scheduler parameters for processes. Both commands use the following pcparms structure to send arguments or receive return values:

```

typedef struct pcparms {
    id_t      pc_cid;          /* process class */
    long     pc_clparms[PC_CLPARMSZ]; /* class-specific */
} pcparms_t;

```

The following function, used in an earlier program example, uses PC_GETPARMS to obtain the scheduler class ID of a process:

```

/*
 * Return scheduler class ID of process with ID pid.
 */
getclassID (pid)
    id_t      pid;
{
    pcparms_t pcparms;

    pcparms.pc_cid = PC_CLNULL;
    if (priocntl(P_PID, pid, PC_GETPARMS, &pcparms) == -1){
        return (-1);
    }
    return (pcparms.pc_cid);
}

```

For the RT class, pc_clparms contains an rtparms structure. The rtparms structure holds scheduler parameters specific to the RT class.

```

typedef struct rtparms {
    short     rt_pri;        /* realtime priority */
    ulong     rt_tqsecs;    /* seconds in time quantum */
    long      rt_tqnsecs;   /* additional nsecs in quantum */
} rtparms_t;

```

In this structure, rt_tqnsecs is a multiple of the nanoseconds it gives another p

For the TS class, parameter spec

```

typedef struct {
    short
    short
} tsparms_t

```

In this structure, priority, and t set for itself wi

The PC_GETPA The return val returned in the arguments to p PC_CLNULL or

Table 15-5 Return

Number Processes Se by idtype a
1
More than 1

If idtype and process, prio than one proce specific criteria

- idtype and
- idtype and
- idtype and

In this structure, `rt_pri` is the realtime priority, `rt_tqsecs` is the number of seconds, and `rt_tqnsecs` is the number of additional nanoseconds in a time slice rounded up to the next multiple of the system clock's resolution. The sum of `rt_tqsecs` seconds and `rt_tqnsecs` nanoseconds is the interval a process can use the CPU without sleeping before the scheduler gives another process a chance at the CPU.

For the TS class, `pc_clparms` contains a `tsparms` structure which holds the scheduler parameter specific to the timesharing class.

```
typedef struct tsparms {
    short  ts_uprilim;      /* user priority limit */
    short  ts_upri;        /* user priority */
} tsparms_t;
```

In this structure, `ts_upri` is the user priority, the user-controlled component of a time-sharing priority, and `ts_uprilim` is the user priority limit, the maximum user priority a process can set for itself without being super-user.

The `PC_GETPARMS` command retrieves the scheduler class and parameters of a single process. The return value of `priocntl(2)` is the process ID of the process whose parameters are returned in the `pcparms` structure. The process chosen depends on the `idtype` and `id` arguments to `priocntl(2)` and on the value of `pcparms.pc_cid`, which contains `PC_CLNULL` or a class ID returned by `PC_GETCID`.

Table 15-5 Return Values for `PC_GETPARMS`

Number of Processes Selected by <code>idtype</code> and <code>id</code>	<code>pc_cid</code>		
	RT class ID	TS class ID	<code>PC_CLNULL</code>
1	RT parameters of process selected	TS parameters of process selected	class and parameters of process selected
More than 1	RT parameters of highest priority RT process	TS parameters of process with highest user priority	(error)

If `idtype` and `id` select a single process and `pc_cid` does not conflict with the class of that process, `priocntl(2)` returns the scheduler parameters of the process. If they select more than one process in a single scheduler class, `priocntl(2)` returns parameters using class-specific criteria as shown in Table 15-5. `priocntl(2)` returns an error in the following cases:

- `idtype` and `id` select one or more processes and none is in the class specified by `pc_cid`.
- `idtype` and `id` select more than one process and `pc_cid` is `PC_CLNULL`.
- `idtype` and `id` select no processes.

RT_TQINF specifies an infinite time slice. RT_TQDEF specifies the default time slice configured for the realtime priority being set with the PC_SETPARMS call. RT_NOCHANGE specifies no change from the current time slice; this value is useful, for example, when you change process priority but do not want to change the time slice. You can also use RT_NOCHANGE in the `rt_pri` field to change a time slice without changing the priority.

The `prIOCtlSet(2)` Function

The `prIOCtlSet(2)` function changes scheduler parameters of a set of processes, just like `prIOCtl(2)`. The `prIOCtlSet(2)` function also has the same command set as `prIOCtl(2)`; the `cmd` and `arg` input arguments are the same. But, while `prIOCtl(2)` applies to a set of processes specified by a single `idtype/id` pair, `prIOCtlSet(2)` applies to a set of processes that results from a logical combination of two `idtype/id` pairs. The following is the format of the `prIOCtlSet(2)` function:

```
#include <sys/types.h>
#include <sys/procset.h>
#include <sys/prIOCtl.h>
#include <sys/rtpriocntl.h>
#include <sys/tspriocntl.h>

long prIOCtlSet(procset_t *psp, int cmd, /* arg */);
```

The input argument `psp` points to a `procset` structure that specifies the two `idtype/id` pairs and the logical operation to perform. This structure is defined in `procset.h`.

```
typedef struct procset {
    idop_t p_op;          /* operator connecting */
                        /* left and right sets */

    /* left set: */
    idtype_t p_lidtype;  /* left ID type */
    id_t      p_lid;     /* left ID */

    /* right set: */
    idtype_t p_ridtype;  /* right ID type */
    id_t      p_rid;     /* right ID */
} procset_t;
```

The `p_lidtype` and `p_lid` parameters specify the ID type and ID of one ("left") set of processes; `p_ridtype` and `p_rid` specify the ID type and ID of a second ("right") set of processes. `p_op` specifies the operation to perform on the two sets of processes to get the set of processes `prIOCtlSet(2)` will act upon.