

\$SPAD/src/lib openpty.c

The Axiom Team

July 28, 2014

**Abstract**

## Contents

|          |                         |          |
|----------|-------------------------|----------|
| <b>1</b> | <b>Overview</b>         | <b>3</b> |
| <b>2</b> | <b>include files</b>    | <b>4</b> |
| <b>3</b> | <b>openpty</b>          | <b>4</b> |
| <b>4</b> | <b>makeNextPtyNames</b> | <b>7</b> |
| <b>5</b> | <b>License</b>          | <b>7</b> |

## 1 Overview

The main function is `ptyopen`. It simply opens up both sides of a pseudo-terminal. It uses and saves the pathnames for the devices which were actually opened.

If it fails it simply exits the program.

```
ptyopen(controller, server, controllerPath, serverPath)
int *controller;    The file descriptor for controller side
int *server;        The file descriptor for the server side
char *controllerPath; actually , this is not used anywhere
                    on return and can be taken out of the
                    call sequence
char *serverPath;
```

The path name vars should be declared of size 11 or more

The device `/dev/ptmx` is the pseudo-terminal master device. The device `/dev/pts` is the pseudo-terminal slave device.

The file `/dev/ptmx` is a character file with a major number of 5 and a minor number of 2, usually of mode 0666 and owner.group of root.root. It is used to create a pseudo-terminal master and slave pair.

When a process opens `/dev/ptmx`, it gets a file descriptor for a pseudo-terminal master PTM, and a pseudo-terminal slave PTS device is created in the `/dev/pts` directory. Each file descriptor obtained by opening `/dev/ptmx` is an independent PTM with its own associated PTS, whose path can be found by passing the descriptor to `ptsname`.

Before opening the pseudo-terminal slave, you must pass the master's file descriptor to `grantpt` and `unlockpt`.

Once both the pseudo-terminal master and slave are open, the slave provides processes with an interface that is identical to that of a real terminal.

Data written to the slave is presented on the master descriptor as input. Data written to the master is presented to the slave as input.

In practice, pseudo-terminals are used for implementing terminal emulators such as `xterm`, in which data read from the pseudo-terminal master is interpreted by the application in the same way a real terminal would interpret the data, and for implementing remote login programs such as `sshd`, in which data read from the pseudo-terminal master is sent across the network to a client program that is connected to a terminal or terminal emulator.

Pseudo-terminals can also be used to send input to programs that normally refuse to read input from pipes (such as `su`) and `passwd`.

The Linux support for the pseudo-terminals (known as Unix98 pty naming) is done using the `devpts` filesystem, that should be mounted on `/dev/pts`.

Before this Unix98 scheme, master ptys were called `/dev/ptyp0`, ..., and slave ptys `/dev/ttyp0`, ... and one to preallocate a lot of device nodes./cite1

## 2 include files

```
— * —

#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>
#include <fcntl.h>
#include <string.h>

#if defined(SUN40S5platform) || defined(HP10platform)
#include <stropts.h>
#endif

#include "openpty.h1"
```

## 3 openpty

```
— * —

int
ptyopen(int *controller,int * server, char *controllerPath,char * serverPath)
{
#if defined(SUNplatform) ||\
defined (HP9platform) ||\
defined(RTplatform) ||\
defined(AIX370platform) ||\
defined(BSDplatform)
    int looking = 1, i;
    int oflag = O_RDWR;                /* flag for opening the pty */

    for (i = 0; looking && i < 1000; i++) {
        makeNextPtyNames(controllerPath, serverPath);
        if (access(controllerPath, 6) != 0) continue;
        *controller = open(controllerPath, oflag, 0);
        if (*controller >= 0) {
            *server = open(serverPath, oflag, 0);
            if (*server > 0)
                looking = 0;
            else
                close(*controller);
        }
    }
}
```

```

    if (looking) {
        fprintf(stderr, "Couldn't find a free pty.\n");
        exit(-1);
    }
    return (*controller);
#endif
#if defined RIOPlatform
    int fdm,fds;
    char *slavename;
    /* open master */
    if ((fdm=open("/dev/ptc",O_RDWR))<0)
        perror("ptyopen failed to open /dev/ptc");
    else {
        /* get slave name */
        if((slavename = ttyname(fdm))==0)
            perror("ptyopen failed to get the slave device name");
        /* open slave */
        if ((fds = open(slavename, O_RDWR)) < 0 )
            perror("ptyopen: Failed to open slave");
        strcpy(serverPath,slavename);
        *controller=fdm;
        *server=fds;
    }
    return(fdm);
#endif

```

---

Note that since we have no other information we are adding the MACOSX-platform variable to the list everywhere we find LINUXplatform. This may not be correct but we have no way to know yet. We have also added the BSDplatform variable. MAC OSX is some variant of BSD. These should probably be merged but we cannot yet prove that.

---

```

— * —

#if defined(SUN40S5platform) ||\
    defined(ALPHAplatform) ||\
    defined(HP10platform) ||\
    defined(LINUXplatform) ||\
    defined(MACOSXplatform) ||\
    defined(BSDplatform)

extern int grantpt(int);
extern int unlockpt(int);
extern char* ptsname(int);
    int fdm,fds;
    char *slavename;

    /* open master */

```

```

if ((fdm = open("/dev/ptmx", O_RDWR)) < 0 )
    perror("ptyopen: Failed to open /dev/ptmx");
else {
    /* change permission of slave */
    if (grantpt(fdm) < 0)
        perror("ptyopen: Failed to grant access to slave device");
    /* unlock slave */
    if (unlockpt(fdm) < 0)
        perror("ptyopen: Failed to unlock master/slave pair");
    /* get name of slave */
    if ((slavename = ptsname(fdm)) == NULL)
        perror("ptyopen: Failed to get name of slave device");
    /* open slave */
    if ((fds = open(slavename, O_RDWR)) < 0 )
        perror("ptyopen: Failed to open slave");
    else {
#ifdef defined(SUN40S5platform) || defined(HP10platform)
        /* push ptem */
        if (ioctl(fds, I_PUSH, "ptem") < 0)
            perror("ptyopen: Failed to push ptem");
        /* push ldterm */
        if (ioctl(fds, I_PUSH, "ldterm") < 0)
            perror("ptyopen: Failed to push ldterm");
#endif
        strcpy(serverPath,slavename);
        *controller=fdm;
        *server=fds;
    }
}
return(fdm);
#endif
#ifdef defined SGIplatform
char *fds;
fds = _getpty(controller, O_RDWR|O_NDELAY, 0600, 0);
strcpy(serverPath,fds);
if (0 == serverPath)
    return(-1);
if (0 > (*server = open(serverPath,O_RDWR))) {
    (void) close(*controller);
    return(-1);
}
return (*controller);
#endif
}

```

---

Prior to using the Unix 98 pty naming scheme the naming scheme used 16 ptyp/ttyp names, ttyp0-ttypF (where F is a hex number). Later this was

extended to ttyq0-ttyqF and so on, eventually wrapping around to ttya0-ttyaF. Linux also allows larger numbers such as ttypNNN.[2]

## 4 makeNextPtyNames

```

    ____ * ____

void
makeNextPtyNames(char *cont, char * serv)
{
#ifdef AIX370platform
static int channelNo = 0;
sprintf(cont, "/dev/ptyp%02x", channelNo);
sprintf(serv, "/dev/ttyp%02x", channelNo);
channelNo++;
#endif

    _____

    See the note above about the MACOS platform change.

    ____ * ____

#if defined(SUNplatform) ||\
    defined(HP9platform) ||\
    defined(LINUXplatform) ||\
    defined(MACOSXplatform) ||\
    defined(BSDplatform)
static int channelNo = 0;
static char group[] = "pqrstuvwxyzPQRST";
static int groupNo = 0;

sprintf(cont, "/dev/pty%c%x", group[groupNo], channelNo);
sprintf(serv, "/dev/tty%c%x", group[groupNo], channelNo);
channelNo++;
/* try next */
if (channelNo == 16) {
/* move to new group */
channelNo = 0;
groupNo++;
if (groupNo == 16) groupNo = 0;
/* recycle */
}
#endif
}

    _____

```

## 5 License

```
/*
```

Copyright (c) 1991-2002, The Numerical Algorithms Group Ltd.  
All rights reserved.

Redistribution and use in source and binary forms, with or without  
modification, are permitted provided that the following conditions are  
met:

- Redistributions of source code must retain the above copyright  
notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright  
notice, this list of conditions and the following disclaimer in  
the documentation and/or other materials provided with the  
distribution.
- Neither the name of The Numerical Algorithms Group Ltd. nor the  
names of its contributors may be used to endorse or promote products  
derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS  
IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED  
TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A  
PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER  
OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,  
EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,  
PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR  
PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF  
LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING  
NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS  
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

\*/



## References

- [1] `ptmx(4)` - Linux man page  
<http://linux.die.net/man/4/ptmx>
- [2] Text Terminal HOWTO  
<http://www.linux.org/docs/ldp/howto/Text-Terminal-HOWTO-7.html>