

Perl / perl5 Public[Code](#) [Issues](#) 2.2k [Pull requests](#) 157 [Actions](#) [Projects](#) [Wiki](#) [Security](#)[New issue](#)

# Spawning threads with open directory handles causes a crash #10387

✓ Closed

Labels

hasCoreDumphasPatchtype-ithreads

p5pRT opened on May 18, 2010

Migrated from [rt.perl.org#75154](https://rt.perl.org/rt/Ticket/Display?id=75154) (status was 'resolved')

Searchable as RT75154\$



p5pRT on May 18, 2010

Author**From [alexford@live.com](mailto:alexford@live.com)****Created by [alexford@live.com](mailto:alexford@live.com)**

Quick summary: if you do an opendir() and then spawn threads, perl will crash.

I've included a brief test case below. The output is something like this:"ok 1Free to wrong pool 3221910 not 3b8580."

```
#!/usr/bin/perluse threads;
```

```
do { my $thread = async { 42 }; opendir my $dir, '.'; $thread->join;  
    print "ok 1\n";};
```

```
do { opendir my $dir, '.'; threads::join(async { 42 });  
    print "ok 2\n";};
```

[▶ Perl Info](#)

p5pRT on May 18, 2010

Author

**From @tsee**

Hi Alex,

Alexander Ford wrote:

```
# New Ticket Created by Alexander Ford
# Please include the string: [perl #75154]
# in the subject line of all future correspondence about this issue.
# <URL: http://rt.perl.org/rt3/Ticket/Display.html?id=75154 >
```

Quick summary: if you do an opendir() and then spawn threads, perl will crash.

I've included a brief test case below.

The output is something like this:

```
"ok 1Free to wrong pool 3221910 not 3b8580."
```

```
#!/usr/bin/perluse threads;
do { my $thread = async { 42 }; opendir my $dir, '.'; $thread->join;
print "ok 1\n";};
do { opendir my $dir, '.'; threads::join(async { 42 });
print "ok 2\n";};
```

I only had a system perl 5.10.0 with a recent threads.pm from CPAN handy. There, I get a crash (double free or corruption) only after "ok 2". Since it doesn't reach a statement outside the second do{} block, I assume it's happening during the closing of the dir handle or respectively the freeing of the \$dir variable.

Best regards,  
Steffen



p5pRT on May 18, 2010

Author



The RT System itself - Status changed from 'new' to 'open'



p5pRT on May 24, 2010

Author

**From @jdhedden**

Steffen, Alex,

What version of threads are you using?

```
perl -Mthreads -e 'print $threads::VERSION, "\n"'
```

I tried your test with threads 1.77 on 5.8.8/9, 5.10.0/1, 5.12.0/1 and blead, and did not get a failure.

If you can upgrade to threads 1.77 and test again, it may be that this problem has already been fixed.

Thanks,  
Jerry

On Tue, May 18, 2010 at 05:52, Steffen Mueller <smueller@cpan.org> wrote:

Hi Alex,

Alexander Ford wrote:

```
# New Ticket Created by Alexander Ford # Please include the string:
[perl #75154]
# in the subject line of all future correspondence about this issue. #
<URL: http://rt.perl.org/rt3/Ticket/Display.html?id=75154 >
```

Quick summary: if you do an opendir() and then spawn threads, perl will crash.

I've included a brief test case below.

The output is something like this:

```
"ok 1Free to wrong pool 3221910 not 3b8580."
```

```
#!/usr/bin/perluse threads;
do { my $thread = async { 42 }; opendir my $dir, '.';
    $thread->join;
    print "ok 1\n";};
do { opendir my $dir, '.'; threads::join(async { 42 });
    print "ok 2\n";};
```

I only had a system perl 5.10.0 with a recent threads.pm from CPAN handy. There, I get a crash (double free or corruption) only after "ok 2". Since it doesn't reach a statement outside the second do{} block, I assume it's happening during the closing of the dir handle or respectively the freeing of the \$dir variable.

Best regards,  
Steffen



p5pRT on May 25, 2010

Author ...

**From @cpansprout**

This is the result of this function in sv.c:

```
DIR *
Perl_dirp_dup(pTHX_ DIR *const dp)
{
    PERL_UNUSED_CONTEXT;
    if (!dp)
        return (DIR*)NULL;
    /* XXX TODO */
    return dp;
}
```

Unfortunately, I don't know how to duplicate a directory handle properly.

The attached patch (not to be applied; just an example) makes a copy using dup2, but the resulting dir handles share the same iterator (i.e., the current behaviour, but without the crash). The code works on Mac OS X, but I doubt it's portable. I don't know how to make it work on other systems.

I can see three ultimate solutions (in order of preference):

- a) Properly duplicate the dir handle so we have a new one with its own iterator (I can't find a fopendir function anywhere)
- b) Simple duplicate the dir handle as in my example
- c) Don't copy dir handles into threads (return (DIR\*)NULL; I tried that and it works)

If b or c happens, I think this needs to be documented in threads.pm. We may have to choose between different items in the list at configure-time, based on OS.



p5pRT on May 25, 2010

Author



**From @cpansprout**

► Inline Patch



p5pRT on Aug 16, 2010

Author



**From @tsee**

Hi Jerry,

Jerry D. Hedden wrote:

What version of threads are you using?

```
perl \-Mthreads \-e 'print $threads&#8203;::VERSION\, "\\n"'
```



I tried your test with threads 1.77 on 5.8.8/9, 5.10.0/1, 5.12.0/1 and blead, and did not get a failure.

If you can upgrade to threads 1.77 and test again, it may be that this problem has already been fixed.

It was threads 1.75. Installing threads 1.77 from CPAN into the ubuntu system 5.10 results in threads 1.77 being shadowed by the old core 1.75. My currently handy copy of bleadperl (static build, so YMMV) gives me this backtrace:

```
#0 0x00007ffff6ba04b5 in *__GI_raise (sig=<value optimized out>) at
../nptl/sysdeps/unix/sysv/linux/raise.c:64
#1 0x00007ffff6ba3f50 in *__GI_abort () at abort.c:92
#2 0x00007ffff6bd91b7 in __libc_message (do_abort=<value optimized
out>, fmt=<value optimized out>) at
../sysdeps/unix/sysv/linux/libc_fatal.c:189
#3 0x00007ffff6be32f6 in malloc_printerr (action=3, str=0x7ffff6ca4cf0
"double free or corruption (!prev)", ptr=<value optimized out>) at
malloc.c:6217
#4 0x00007ffff6be7c6c in *__GI___libc_free (mem=<value optimized out>)
at malloc.c:3716
#5 0x00007ffff6c0d0bd in __closedir (dirp=0x26fa) at
../sysdeps/unix/closedir.c:52
#6 0x000000000086d1b1 in Perl_sv_clear (my_perl=0xd96010, sv=0xd99a00)
at sv.c:5842
#7 0x000000000086e5cc in Perl_sv_free2 (my_perl=0xd96010, sv=0xd99a00)
at sv.c:6064
#8 0x0000000000720d67 in Perl_gp_free (my_perl=0xd96010, gv=0xe537f8)
at gv.c:1644
#9 0x000000000086d862 in Perl_sv_clear (my_perl=0xd96010, sv=0xe537f8)
at sv.c:5887
#10 0x000000000086e5cc in Perl_sv_free2 (my_perl=0xd96010, sv=0xe537f8)
at sv.c:6064
#11 0x0000000000886f4f in Perl_sv_unref_flags (my_perl=0xd96010,
ref=0xdf5b68, flags=1) at sv.c:9074
#12 0x0000000000862cca in Perl_sv_force_normal_flags (my_perl=0xd96010,
sv=0xdf5b68, flags=1) at sv.c:4578
#13 0x00000000008ea7e7 in Perl_leave_scope (my_perl=0xd96010, base=2) at
scope.c:906
#14 0x00000000008e52f1 in Perl_pop_scope (my_perl=0xd96010) at scope.c:110
#15 0x0000000000827231 in Perl_pp_leave (my_perl=0xd96010) at pp_hot.c:1919
#16 0x00000000007cfd1 in Perl_runops_debug (my_perl=0xd96010) at
dump.c:2112
#17 0x0000000000701957 in S_run_body (my_perl=0xd96010, oldscope=1) at
perl.c:2312
#18 0x0000000000700b7f in perl_run (my_perl=0xd96010) at perl.c:2236
#19 0x000000000040769d in main (argc=2, argv=0x7ffffffe0a8,
env=0x7ffffffe0c0) at perlmain.c:117
```

Sorry for the delayed reply!

Cheers,  
Steffen



p5pRT on Sep 1, 2010

Author



From [sinantrop@gmail.com](mailto:sinantrop@gmail.com)

This is a bug report for perl from sinantrop@gmail.com,  
generated with the help of perlbug 1.39 running under perl 5.12.1.

This program crashes perl on windows with those messages:  
Free to wrong pool 18247e8 not 245a60 at 1.pl line 11.  
Free to wrong pool 1859c50 not 245a60 at 1.pl line 11.

```
sub ListDir
{
    my ($dir, $proc)=@_;

    local *DH;
    opendir(DH,$dir);
    while(my $fn=readdir(DH))
    {
        &$proc($fn, $dir);
    }
    closedir(DH);
}

ListDir(".", sub
{
    my ($fn, $dir)=@_;
    print "$dir/$fn\n";

    fork;
});
```

---

Flags:

category=core  
severity=medium

---

Site configuration information for perl 5.12.1:

Configured by SYSTEM at Fri May 14 00:24:46 2010.

Summary of my perl5 (revision 5 version 12 subversion 1) configuration:

Platform:

osname=MSWin32, osvers=5.00, archname=MSWin32-x86-multi-thread  
 uname=""  
 config\_args='undef'  
 hint=recommended, useposix=true, d\_sigaction=undef  
 useithreads=define, usemultiplicity=define  
 useperlio=define, d\_sfio=undef, uselargefiles=define, usesocks=undef  
 use64bitint=undef, use64bitall=undef, uselongdouble=undef  
 usemymalloc=n, bincompat5005=undef

Compiler:

cc='cl', ccflags='-nologo -GF -W3 -MD -Zi -DNDEBUG -O1 -DWIN32  
 -D\_CONSOLE -DNO\_STRICT -DHAVE\_DES\_FCRYPT -DUSE\_SITECUSTOMIZE  
 -DPERL\_IMPLICIT\_CONTEXT -DPERL\_IMPLICIT\_SYS -DUSE\_PERLIO  
 -DPERL\_MSVCRT\_READFIX',  
 optimize='-MD -Zi -DNDEBUG -O1',  
 cppflags='-DWIN32'  
 ccversion='12.00.8804', gccversion="", gccosandvers=""  
 intsize=4, longsize=4, ptrsize=4, doublesize=8, byteorder=1234  
 d\_longlong=undef, longlongsize=8, d\_longdbl=define, longdblsize=8  
 ivtype='long', ivsize=4, nvtype='double', nvsize=8, Off\_t='\_\_int64',  
 lseeksize=8

alignbytes=8, prototype=define

Linker and Libraries:

ld='link', ldflags='-nologo -nodefaultlib -debug -opt:ref,icf  
 -libpath:"C:\Perl\lib\CORE" -machine:x86'  
 libpth=\lib  
 libs= oldnames.lib kernel32.lib user32.lib gdi32.lib winspool.lib  
 comdlg32.lib advapi32.lib shell32.lib ole32.lib oleaut32.lib  
 netapi32.lib uuid.lib ws2\_32.lib mpr.lib winmm.lib version.lib  
 odbcc32.lib odbccp32.lib comctl32.lib msvcrt.lib  
 perllibs= oldnames.lib kernel32.lib user32.lib gdi32.lib  
 winspool.lib comdlg32.lib advapi32.lib shell32.lib ole32.lib  
 oleaut32.lib netapi32.lib uuid.lib ws2\_32.lib mpr.lib winmm.lib  
 version.lib odbcc32.lib odbccp32.lib comctl32.lib msvcrt.lib  
 libc=msvcrt.lib, so=dll, useshrplib=true, libperl=perl512.lib  
 gnulibc\_version=""

Dynamic Linking:

dlsrc=dl\_win32.xs, dlext=dll, d\_dlsymun=undef, ccdlflags=' '  
 cccdlflags=' ', lddlflags='-dll -nologo -nodefaultlib -debug  
 -opt:ref,icf -libpath:"C:\Perl\lib\CORE" -machine:x86'

Locally applied patches:

ACTIVEPERL\_LOCAL\_PATCHES\_ENTRY

[d956618](#) Make Term::ReadLine::findConsole fall back to STDIN if

/dev/tty can't be opened

[321e50c](#) Escape patch strings before embedding them in patchlevel.h

---

@INC for perl 5.12.1:

c:/Perl/site/lib

c:/Perl/lib

.

---

Environment for perl 5.12.1:

HOME (unset)

LANG (unset)

LANGUAGE (unset)

LD\_LIBRARY\_PATH (unset)

LOGDIR (unset)

PATH=C:\WINDOWS\system32;C:\WINDOWS;C:\WINDOWS\System32\Wbem;c:\perl\bin

PERL\_BADLANG (unset)

SHELL (unset)



p5pRT on Sep 1, 2010

Author



## From @chorny

On Wed Sep 01 10:03:57 2010, strop wrote:

This is a bug report for perl from sinantrop@gmail.com, generated with the help of perlbug 1.39 running under perl 5.12.1.

This program crashes perl on windows with those messages:

Free to wrong pool 18247e8 not 245a60 at 1.pl line 11.

Free to wrong pool 1859c50 not 245a60 at 1.pl line 11.

```
sub ListDir
```

```
{
```

```
my ($dir, $proc)=@_;
```

```

local *DH;
opendir(DH,$dir);
while(my $fn=readdir(DH))
{
    &$proc($fn, $dir);
}
closedir(DH);
}

ListDir(".", sub
{
    my ($fn, $dir)=@_;
    print "$dir/$fn\n";

    fork;
});
---
```

I'm not sure that it is a good idea to mix dirhandles and fork/threads.

This program works correctly:

```

sub ListDir
{
    my ($dir, $proc)=@_;

    opendir(my $dh,$dir);
    my @files=readdir($dh);
    closedir($dh);
    foreach my $fn (@files)
    {
        $proc->($fn, $dir);
    }
}

ListDir(".", sub
{
    my ($fn, $dir)=@_;
    print "$dir/$fn\n";

    fork;
});
```

It's result are meaningless though - for last file \$proc will be executed 2^N times, where N is numner of files.

--

Alexandr Ciornii, <http://chorny.net>



p5pRT on Sep 1, 2010

Author



The RT System itself - Status changed from 'new' to 'open'



p5pRT on Sep 1, 2010

Author



## From @jdhedden

On Wed Sep 01 10:03:57 2010, strop wrote:

This is a bug report for perl from sinantrop@gmail.com, generated with the help of perlbug 1.39 running under perl 5.12.1.

This program crashes perl on windows with those messages:

Free to wrong pool 18247e8 not 245a60 at 1.pl line 11.

Free to wrong pool 1859c50 not 245a60 at 1.pl line 11.

```
sub ListDir
{
my ($dir, $proc)=@_;

local *DH;
opendir(DH,$dir);
while(my $fn=readdir(DH))
{
&$proc($fn, $dir);
}
closedir(DH);
}
```

```
ListDir(".", sub
{
my ($fn, $dir)=@_;
print "$dir/$fn\n";
```

```
fork;
});
```

```
---
```

Flags:

category=core

severity=medium

```
---
```

Site configuration information for perl 5.12.1:

Configured by SYSTEM at Fri May 14 00:24:46 2010.

Summary of my perl5 (revision 5 version 12 subversion 1) configuration:

Platform:

osname=MSWin32, osvers=5.00, archname=MSWin32-x86-multi-thread  
uname=""  
config\_args='undef'  
hint=recommended, useposix=true, d\_sigaction=undef  
useithreads=define, usemultiplicity=define  
useperlio=define, d\_sfio=undef, uselargefiles=define, usesocks=undef  
use64bitint=undef, use64bitall=undef, uselongdouble=undef  
usemymalloc=n, bincompat5005=undef

Compiler:

cc='cl', ccflags='-nologo -GF -W3 -MD -Zi -DNDEBUG -O1 -DWIN32  
-D\_CONSOLE -DNO\_STRICT -DHAVE\_DES\_FCRYPT -DUSE\_SITECUSTOMIZE  
-DPERL\_IMPLICIT\_CONTEXT -DPERL\_IMPLICIT\_SYS -DUSE\_PERLIO  
-DPERL\_MSVCRT\_READFIX',  
optimize='-MD -Zi -DNDEBUG -O1',  
cppflags='-DWIN32'  
ccversion='12.00.8804', gccversion="", gccosandvers=""  
intsize=4, longsize=4, ptrsize=4, doublesize=8, byteorder=1234  
d\_longlong=undef, longlongsize=8, d\_longdbl=define, longdblsize=8  
ivtype='long', ivsize=4, nvtype='double', nvsize=8, Off\_t='\_\_int64',  
lseeksize=8  
alignbytes=8, prototype=define

Linker and Libraries:

ld='link', ldflags='-nologo -nodefaultlib -debug -opt:ref,icf  
-libpath:"C:\Per\lib\CORE" -machine:x86'  
libpth=\lib  
libs= oldnames.lib kernel32.lib user32.lib gdi32.lib winspool.lib  
comdlg32.lib advapi32.lib shell32.lib ole32.lib oleaut32.lib  
netapi32.lib uuid.lib ws2\_32.lib mpr.lib winmm.lib version.lib  
odbc32.lib odbccp32.lib comctl32.lib msvcrt.lib  
perllibs= oldnames.lib kernel32.lib user32.lib gdi32.lib  
winspool.lib comdlg32.lib advapi32.lib shell32.lib ole32.lib  
oleaut32.lib netapi32.lib uuid.lib ws2\_32.lib mpr.lib winmm.lib  
version.lib odbc32.lib odbccp32.lib comctl32.lib msvcrt.lib  
libc=msvcrt.lib, so=dll, useshrplib=true, libperl=perl512.lib  
gnulibc\_version=""

Dynamic Linking:

dlsrc=dl\_win32.xs, dlext=dll, d\_dlsymun=undef, ccdlflags=' '  
cccdlflags=' ', lddlflags='-dll -nologo -nodefaultlib -debug  
-opt:ref,icf -libpath:"C:\Per\lib\CORE" -machine:x86'



```
sub ListDir
{
my ($dir, $proc)=@_;

local *DH;
opendir(DH,$dir);
while(my $fn=readdir(DH))
{
&$proc($fn, $dir);
}
closedir(DH);
}

ListDir(".", sub
{
my ($fn, $dir)=@_;
print "$dir/$fn\n";

fork;
});
---
```



p5pRT on Sep 19, 2010

Author



## From @cpansprout

On Tue May 25 01:42:10 2010, sprout wrote:

This is the result of this function in sv.c:

```
DIR *
Perl_dirp_dup(pTHX_ DIR *const dp)
{
PERL_UNUSED_CONTEXT;
if (!dp)
return (DIR*)NULL;
/* XXX TODO */
return dp;
}
```

Unfortunately, I don't know how to duplicate a directory handle properly.

The attached patch (not to be applied; just an example) makes a copy using dup2, but the resulting dir handles share the same iterator (i.e., the current behaviour, but without the crash). The code works on Mac OS X, but I doubt it's portable. I don't know how to make it work on other systems.

I can see three ultimate solutions (in order of preference):

- a) Properly duplicate the dir handle so we have a new one with its own iterator (I can't find a fopendir function anywhere)
- b) Simple duplicate the dir handle as in my example
- c) Don't copy dir handles into threads (return (DIR\*)NULL; I tried that and it works)

If b or c happens, I think this needs to be documented in threads.pm. We may have to choose between different items in the list at configure-time, based on OS.

Attached is a patch to fix this. On systems with fchdir, it dwims (a). On other systems the dir handle is simply not passed to the thread (c).



p5pRT on Sep 19, 2010

Author ...

### From @cpansprout

From: Father Chrysostomos <sprout@cpan.org>

[perl #75174] Clone dir handles

On systems that support fchdir, use it to clone dir handles.

On other systems, at least for now, don't give the new thread a copy of the handle. This is not ideal, but better than crashing.

► Inline Patch



p5pRT on Sep 21, 2010

Author ...

### From @jdhedden

This patch is written to rely on dirent->d\_namlen which doesn't always exist. (I tried it on Cygwin and got build failures). I tried replacing occurrences of "dirent->d\_namlen" with "strlen(dirent->d\_name)+1", and it built and tested successfully.

Also, there's a build warning about 'len' possibly being uninitialized, so initializing it to something like -1 should work.

On Sun, Sep 19, 2010 at 16:10, Father Chrysostomos via RT <perlbug-followup@perl.org> wrote:

On Tue May 25 01:42:10 2010, sprout wrote:

This is the result of this function in sv.c:

```
DIR *
Perl_dirp_dup(pTHX_ DIR *const dp)
{
    PERL_UNUSED_CONTEXT;
    if (!dp)
        return (DIR*)NULL;
    /* XXX TODO */
    return dp;
}
```

Unfortunately, I don't know how to duplicate a directory handle properly.

The attached patch (not to be applied; just an example) makes a copy using dup2, but the resulting dir handles share the same iterator (i.e., the current behaviour, but without the crash). The code works on Mac OS X, but I doubt it's portable. I don't know how to make it work on other systems.

I can see three ultimate solutions (in order of preference):

- a) Properly duplicate the dir handle so we have a new one with its own iterator (I can't find a fopendir function anywhere)
- b) Simple duplicate the dir handle as in my example
- c) Don't copy dir handles into threads (return (DIR\*)NULL; I tried that and it works)

If b or c happens, I think this needs to be documented in threads.pm. We may have to choose between different items in the list at configure-time, based on OS.

Attached is a patch to fix this. On systems with fchdir, it dwims (a). On other systems the dir handle is simply not passed to the thread (c).

From: Father Chrysostomos <sprout@cpan.org>

[perl #75174] Clone dir handles

On systems that support fchdir, use it to clone dir handles.

On other systems, at least for now, don't give the new thread a copy of the handle. This is not ideal, but better than crashing.

```

diff -up bleed/MANIFEST bleed-75154-dirdup/MANIFEST
--- bleed/MANIFEST 2010-09-11 14:19:12.000000000 -0700
+++ bleed-75154-dirdup/MANIFEST 2010-09-17 12:49:00.000000000 -0700
@@ -4635,6 +4635,7 @@ t/op/symbolcache.t          See if undef/delete
t/op/sysio.t          See if sysread and syswrite work
t/op/taint.t          See if tainting works
t/op/threads_create.pl  Ancillary file for t/op/threads.t
+t/op/threads-dirh.t    Test interaction of threads and dir handles
t/op/threads.t        Misc. tests for perl features with threads
t/op/tiearray.t       See if tie for arrays works
t/op/tie_fetch_count.t See if FETCH is only called once on tied variables
diff -up bleed/sv.c bleed-75154-dirdup/sv.c
--- bleed/sv.c 2010-09-09 12:00:10.000000000 -0700
+++ bleed-75154-dirdup/sv.c 2010-09-18 06:21:19.000000000 -0700
@@ -10849,11 +10849,95 @@ Perl_fp_dup(pTHX_ PerlIO *const fp, cons
DIR *
Perl_dirp_dup(pTHX_ DIR *const dp)
{
+#ifdef HAS_FCHDIR
+  DIR *ret;
+  DIR *pwd;
+  register const Dirent_t *dirent;
+  char smallbuf[256];
+  char *name = NULL;
+  STRLEN len;
+  long pos;
+#endif
+
+  PERL_UNUSED_CONTEXT;
+
+#ifdef HAS_FCHDIR
+  if (!dp)
+    return (DIR*)NULL;
-  /* XXX TODO */
-  return dp;
+  /* look for it in the table first */
+  ret = (DIR*)ptr_table_fetch(PL_ptr_table, dp);
+  if (ret)
+    return ret;
+
+  /* create anew */
+
+  /* open the current directory (so we can switch back) */
+  if (!(pwd = PerlDir_open("."))) return (DIR *)NULL;
+
+  /* chdir to our dir handle and open the present working directory */

```

```

+ if (fchdir(my_dirfd(dp)) < 0 || !(ret = PerlDir_open("."))) {
+   PerlDir_close(pwd);
+   return (DIR *)NULL;
+ }
+ /* Now we should have two dir handles pointing to the same dir. */
+
+ /* Be nice to the calling code and chdir back to where we were. */
+ fchdir(my_dirfd(pwd)); /* If this fails, then what? */
+
+ /* We have no need of the pwd handle any more. */
+ PerlDir_close(pwd);
+
+ /* Iterate once through dp, to get the file name at the current posi-
+   tion. Then step back. */
+ pos = PerlDir_tell(dp);
+ if ((dirent = PerlDir_read(dp))) {
+   len = dirent->d_namlen;
+   if (len <= sizeof smallbuf) name = smallbuf;
+   else Newx(name, len, char);
+   Move(dirent->d_name, name, len, char);
+ }
+ PerlDir_seek(dp, pos);
+
+ /* Iterate through the new dir handle, till we find a file with the
+   right name. */
+ if (!dirent) /* just before the end */
+   for(;;) {
+     pos = PerlDir_tell(ret);
+     if (PerlDir_read(ret)) continue; /* not there yet */
+     PerlDir_seek(ret, pos); /* step back */
+     break;
+   }
+ else {
+   const long pos0 = PerlDir_tell(ret);
+   for(;;) {
+     pos = PerlDir_tell(ret);
+     if ((dirent = PerlDir_read(ret))) {
+       if (len == dirent->d_namlen
+         && memEQ(name, dirent->d_name, len)) {
+         /* found it */
+         PerlDir_seek(ret, pos); /* step back */
+         break;
+       }
+     }
+     /* else we are not there yet; keep iterating */
+   }
+ }
+ else { /* This is not meant to happen. The best we can do is

```

```

+         reset the iterator to the beginning. */
+     PerlDir_seek(ret, pos0);
+     break;
+ }
+ }
+ }
+
+ if (name && name != smallbuf)
+     Safefree(name);
+
+ /* pop it in the pointer table */
+ ptr_table_store(PL_ptr_table, dp, ret);
+
+ return ret;
+
+ #else
+ return (DIR*)NULL;
+
+ #endif
+ }

```

/\* duplicate a typeglob \*/

```

diff -up bleed/dist/threads/lib/threads.pm bleed-75154-dirdup/dist/threads/lib/threads.pm
--- bleed/dist/threads/lib/threads.pm 2010-07-07 07:22:10.000000000 -0700
+++ bleed-75154-dirdup/dist/threads/lib/threads.pm 2010-09-17 01:12:59.000000000 -0700
@@ -1005,6 +1005,16 @@ mutexes that are needed to control funct
For this reason, the use of C<END> blocks in threads is B<strongly>
discouraged.

```

=item Directory handles

+  
+In perl 5.14.0 and higher, if your system does not support the C<fchdir> C  
+function, directory handles will not be copied to new threads. You can use  
+the C<d\_fchdir> variable in L<Config.pm|Config> to determine whether your  
+system supports it.

+  
+In prior perl versions, leaving directory handles open when threads were  
+created could result in crashes or memory corruption.

+  
=item Perl Bugs and the CPAN Version of L<threads>

```

Support for threads extends beyond the code in this module (i.e.,
diff -rNup blead/t/op/threads-dirh.t blead-75154-dirdup/t/op/threads-dirh.t
--- blead/t/op/threads-dirh.t 1969-12-31 16:00:00.000000000 -0800
+++ blead-75154-dirdup/t/op/threads-dirh.t 2010-09-18 06:17:53.000000000 -0700
@@ -0,0 +1,131 @@
+#!perl
+
+# Test interaction of threads and directory handles.
+
+BEGIN {
+  chdir 't' if -d 't';
+  @INC = './lib';
+  require './test.pl';
+  $| = 1;
+
+  require Config;
+  if (!$Config::Config{useithreads}) {
+    print "1..0 # Skip: no ithreads\n";
+    exit 0;
+  }
+  if ($ENV{PERL_CORE_MINITEST}) {
+    print "1..0 # Skip: no dynamic loading on miniperl, no threads\n";
+    exit 0;
+  }
+
+  plan(6);
+}
+
+use strict;
+use warnings;
+use threads;
+use threads::shared;
+use File::Path;
+use File::Spec::Functions qw 'updir catdir';
+use Cwd 'getcwd';
+
+# Basic sanity check: make sure this does not crash
+fresh_perl_is <<'# this is no comment', 'ok', {}, 'crash when duping dirh';
+  use threads;
+  opendir dir, 'op';
+  async{}->join for 1..2;
+  print "ok";
+# this is no comment
+
+my $dir;
+SKIP: {

```

```

+ my $skip = sub {
+   chdir($dir);
+   chdir updir;
+   skip $_[0], 5
+ };
+
+ if(!$Config::Config{d_fchdir}) {
+   $::TODO = 'dir handle cloning currently requires fchdir';
+ }
+
+ my @w :shared; # warnings accumulator
+ local $SIG{__WARN__} = sub { push @w, $_[0] };
+
+ $dir = catdir getcwd(), "thrext$$" . int rand() * 100000;
+
+ rmtree($dir);
+ mkdir($dir);
+
+ # Create a dir structure like this:
+ # $dir
+ # |
+ # `-- toberead
+ #     |
+ #     +---- thrit
+ #     |
+ #     +---- rile
+ #     |
+ #     `---- zor
+
+ chdir($dir);
+ mkdir 'toberead';
+ chdir 'toberead';
+ {open my $fh, ">thrit" or &$skip("Cannot create file thrit")}
+ {open my $fh, ">rile" or &$skip("Cannot create file rile")}
+ {open my $fh, ">zor" or &$skip("Cannot create file zor")}
+ chdir updir;
+
+ # Then test that dir iterators are cloned correctly.
+
+ opendir my $toberead, 'toberead';
+ my $start_pos = telldir $toberead;
+ my @first_2 = (scalar readdir $toberead, scalar readdir $toberead);
+ my @from_thread = @{}; async { [readdir $toberead ] } ->join };
+ my @from_main = readdir $toberead;
+ is join('-', sort @from_thread), join('-', sort @from_main),
+   'dir iterator is copied from one thread to another';

```

```

+ like
+ join('-', '', sort(@first_2, @from_thread), ''),
+ qr/(?<!-rile)-rile-thrit-zor-(?!zor-)/i,
+ 'cloned iterator iterates exactly once over everything not already seen';
+
+ seekdir $toberead, $start_pos;
+ readdir $toberead for 1 .. @first_2+@from_thread;
+ is
+ async { readdir $toberead // 'undef' } ->join, 'undef',
+ 'cloned dir iterator that points to the end of the directory'
+ ;
+
+ # Make sure the cloning code can handle file names longer than 255 chars
+ SKIP: {
+ chdir 'toberead';
+ open my $fh,
+ ">floccipaucinihilopilification-"
+ . "pneumonoultramicroscopicsilicovolcanoconiosis-"
+ . "lopadotemachoselachogaleokraniroleipsanodrimypotrimmatosilphiokarabo"
+ . "melitokatakechymenokichlepikossyphophattoperisteralektryonoptokephal"
+ . "liokinklopeleiolagoiosiraibaphetraganopterygon"
+ or
+ chdir updir,
+ skip("OS does not support long file names (and I mean *long*)", 1);
+ chdir updir;
+ opendir my $dirh, "toberead";
+ my $test_name
+ = "dir iterators can be cloned when the next fn > 255 chars";
+ while() {
+ my $pos = telldir $dirh;
+ my $fn = readdir($dirh);
+ if(!defined $fn) { fail($test_name); last SKIP; }
+ if($fn =~ 'lagoio') {
+ seekdir $dirh, $pos;
+ last;
+ }
+ }
+ is length async { scalar readdir $dirh } ->join, 257, $test_name;
+ }
+
+ is scalar @w, 0, 'no warnings during all that' or diag @w;
+ chdir updir;
+ }
+ rmtree($dir);

```



p5pRT on Sep 26, 2010

Author



## From @cpansprout

On Tue Sep 21 10:59:20 2010, jdhedden wrote:

This patch is written to rely on `dirent->d_namlen` which doesn't always exist. (I tried it on Cygwin and got build failures). I tried replacing occurrences of `"dirent->d_namlen"` with `"strlen(dirent->d_name)+1"`, and it built and tested successfully.

Why +1? `d_namlen` does not include the null.

Does this new patch work for you?



p5pRT on Sep 26, 2010

Author



## From @cpansprout

From: Father Chrysostomos <sprout@cpan.org>

[perl #75174] Clone dir handles

On systems that support `fchdir`, use it to clone dir handles.

On other systems, at least for now, don't give the new thread a copy of the handle. This is not ideal, but better than crashing.

► Inline Patch



p5pRT on Sep 27, 2010

Author



## From @jdhedden

On Sun, Sep 26, 2010 at 19:42, Father Chrysostomos via RT <perlbug-followup@perl.org> wrote:

On Tue Sep 21 10:59:20 2010, jdhedden wrote:

This patch is written to rely on `dirent->d_namlen` which doesn't always exist. (I tried it on Cygwin and got build failures). I tried replacing occurrences of `"dirent->d_namlen"` with `"strlen(dirent->d_name)+1"`, and it built and tested successfully.

Why +1? `d_namlen` does not include the null.

Does this new patch work for you?

Yes, your new patch compiles and tests successfully.



p5pRT on Sep 28, 2010

Author



## From @cpansprout

On Mon Sep 27 13:01:46 2010, jdhedden wrote:

On Sun, Sep 26, 2010 at 19:42, Father Chrysostomos via RT <perlbug-followup@perl.org> wrote:

On Tue Sep 21 10:59:20 2010, jdhedden wrote:

This patch is written to rely on dirent->d\_namlen which doesn't always exist. (I tried it on Cygwin and got build failures). I tried replacing occurrences of "dirent->d\_namlen" with "strlen(dirent->d\_name)+1", and it built and tested successfully.

Why +1? d\_namlen does not include the null.

Does this new patch work for you?

Yes, your new patch compiles and tests successfully.

Applied as 11a11e.



p5pRT on Sep 28, 2010

Author



**@cpansprout** - Status changed from 'open' to 'resolved'



**p5pRT** closed this as completed on Sep 28, 2010



**p5pRT** added **Severity High** **hasCoreDump** **hasPatch** **type-ithreads** on Oct 18, 2019



**vinc17fr** mentioned this on Feb 18, 2025



Thread creation while a directory handle is open does a fchdir, affecting other threads (race condition) #23010



**tonycoz** mentioned this on Feb 24, 2025

[Clone dirhandles without fchdir #23019](#)

[Sign up for free](#) to join this conversation on **GitHub**. Already have an account? [Sign in to comment](#)

## Metadata

### Assignees

No one assigned

### Labels

hasCoreDump

hasPatch

type-ithreads

### Type

No type

### Projects

No projects

### Milestone

No milestone

### Relationships

None yet

### Development

No branches or pull requests

### Participants



