**NAME**

libunwind-ptrace -- ptrace() support in libunwind

**SYNOPSIS**

```
#include <libunwind-ptrace.h>

unw_accessors_t _UPT_accessors;

void *_UPT_create(pid_t);
void _UPT_destroy(void *);

int _UPT_find_proc_info(unw_addr_space_t, unw_word_t, unw_proc_info_t *, int, void *);
void _UPT_put_unwind_info(unw_addr_space_t, unw_proc_info_t *, void *);
int _UPT_get_dyn_info_list_addr(unw_addr_space_t, unw_word_t *, void *);
int _UPT_access_mem(unw_addr_space_t, unw_word_t, unw_word_t *, int, void *);
int _UPT_access_reg(unw_addr_space_t, unw_regnum_t, unw_word_t *, int, void *);
int _UPT_access_fpreg(unw_addr_space_t, unw_regnum_t, unw_fpreg_t *, int, void *);
int _UPT_get_proc_name(unw_addr_space_t, unw_word_t, char *, size_t, unw_word_t *, void *);
int _UPT_resume(unw_addr_space_t, unw_cursor_t *, void *);
```

**DESCRIPTION**

The ptrace(2) system-call makes it possible for a process to gain access to the machine-state and virtual memory of *another* process. With the right set of call-back routines, it is therefore possible to hook up libunwind to another process via ptrace(2). While it's not very difficult to do so directly, libunwind further facilitates this task by providing ready-to-use callbacks for this purpose. The routines and variables implementing this facility use a name-prefix of _UPT, which is stands for ''unwind-via-ptrace''.

An application that wants to use the _UPT-facility first needs to create a new libunwind address-space that represents the target process. This is done by calling unw_create_addr_space(). In many cases, the application will simply want to pass the address of _UPT_accessors as the first argument to this routine. Doing so will ensure that libunwind will be able to properly unwind the target process. However, in special circumstances, an application may prefer to use only portions of the _UPT-facility. For this reason, the individual callback routines (_UPT_find_proc_info(), _UPT_put_unwind_info(), etc.) are also available for direct use. Of course, the addresses of these routines could also be picked up from _UPT_accessors, but doing so would prevent static initialization. Also, when using _UPT_accessors, *all* the callback routines will be linked into the application, even if they are never actually called.

Next, the application can turn on ptrace-mode on the target process, either by forking a new process,

invoking PTRACE_TRACEME, and then starting the target program (via execve(2)), or by directly attaching to an already running process (via PTRACE_ATTACH). Either way, once the process-ID (pid) of the target process is known, a _UPT-info-structure can be created by calling _UPT_create(), passing the pid of the target process as the only argument. The returned void-pointer then needs to be passed as the ‘‘argument’’ pointer (third argument) to unw_init_remote().

The _UPT_resume() routine can be used to resume execution of the target process. It simply invokes ptrace(2) with a command value of PTRACE_CONT.

When the application is done using libunwind on the target process, _UPT_destroy() needs to be called, passing it the void-pointer that was returned by the corresponding call to _UPT_create(). This ensures that all memory and other resources are freed up.

## AVAILABILITY

Since ptrace(2) works within a single machine only, the _UPT-facility by definition is not available in libunwind-versions configured for cross-unwinding.

## THREAD SAFETY

The _UPT-facility assumes that a single _UPT-info structure is never shared between threads. Because of this, no explicit locking is used. As long as only one thread uses a _UPT-info structure at any given time, this facility is thread-safe.

## RETURN VALUE

_UPT_create() may return a NULL pointer if it fails to create the _UPT-info-structure for any reason. For the current implementation, the only reason this call may fail is when the system is out of memory.

## FILES

libunwind-ptrace.h

Headerfile to include when using the interface defined by this library.

**-l**unwind-ptrace **-l**unwind-generic

Linker-switches to add when building a program that uses the functions defined by this library.

## SEE ALSO

execve(2), libunwind(3), ptrace(2)

## AUTHOR

David Mosberger-Tang
Email: **dmosberger@gmail.com**
WWW: **http://www.nongnu.org/libunwind/**.