

**NAME**

rpoll - callback functions for file descriptors and timers

**SYNOPSIS**

```
# include <rpoll.h>
```

```
typedef void (*poll_f)(int fd, int mask, void *arg);
```

```
typedef void (*timer_f)(int tid, void *arg);
```

```
int poll_register(int fd, poll_f func, void *arg, int mask);
```

```
void poll_unregister(int handle);
```

```
int poll_start_timer(u_int msecs, int repeat, timer_f func,  
void *arg);
```

```
void poll_stop_timer(int handle);
```

```
int poll_start_utimer(unsigned long long usecs, int repeat,  
timer_f func, void *arg);
```

```
void poll_dispatch(int wait);
```

**DESCRIPTION**

Many programs need to read from several file descriptors at the same time. Typically in these programs one of **select**(3c) or **poll**(2) is used. These calls are however clumsy to use and the usage of one of these calls is probably not portable to other systems - not all systems support both calls.

The **rpoll**(1) family of functions is designed to overcome these restrictions. They support the well known and understood technique of event driven programing and, in addition to **select**(3c) and **poll**(2) also support timers.

Each event on a file descriptor or each timer event is translated into a call to a user defined callback function. These functions need to be registered. A file descriptor is registered with **poll\_register**. *fd* is the file descriptor to watch, *mask* is an event mask. It may be any combination of **POLL\_IN** to get informed when input on the file descriptor is possible, **POLL\_OUT** to get informed when output is possible or **POLL\_EXCEPT** to get informed when an exceptional condition occurs. An example of an exceptional condition is the arrival of urgent data. (Note, that an end of file condition is signaled via **POLL\_IN**). *func* is the user function to be called and *arg* is a user supplied argument for this function. The callback functions is called with the file descriptor, a mask describing the actual events (from the

set supplied in the registration) and the user argument. **poll\_register** returns a handle, which may be used later to de-register the file descriptor. A file descriptor may be registered more than once, if the function, the user arguments or both differ in the call to **poll\_register**. If *func* and *arg* are the same, then no new registration is done, instead the event mask of the registration is changed to reflect the new mask.

A registered file descriptor may be de-registered by calling **poll\_unregister** with the handle returned by **poll\_register**.

A timer is created with **poll\_start\_timer** or **poll\_start\_utimer**. *msecs* is the number of milliseconds in **poll\_start\_timer** while *usecs* is the number of microseconds in **poll\_start\_utimer**, after which the timer event will be generated. If the functions use the **poll(2)** system call, then *usecs* is rounded to milliseconds and **poll\_start\_timer** is called. *repeat* selects one-shot behavior (if 0) or a repeatable timer (if not 0). A one-shot timer will automatically unregister after expiry. *func* is the user function which will be called with a timer id and the user supplied *arg*. **poll\_start\_timer** and **poll\_start\_utimer** return a timer id, which may be used to cancel the timer with **poll\_stop\_timer**. A one-shot timer should be canceled only if it has not yet fired.

**poll\_dispatch** must be called to actually dispatch events. *wait* is a flag, which should be 0, if only a poll should be done. In this case, the function returns, after polling the registered file descriptors and timers. If *wait* is not 0, **poll\_dispatch** waits until an event occurs. All events are dispatch (i.e. callback functions called) and **poll\_dispatch** returns.

Typical use is:

```
while(1)
    poll_dispatch(1);
```

## SEE ALSO

**poll(2)**, **select(3C)**

## RETURN VALUES

**poll\_register**, **poll\_start\_timer** and **poll\_start\_utimer** return a handle which may be used to unregister the file descriptor or cancel the timer.

Both functions and **poll\_dispatch** call **xrealloc(1)** and can end in **panic(1)**.

## ERRORS

System call or memory allocation errors are fatal and are handle by calling **panic(1)**. The one exception is a return of EINTR from **select(3c)** or **poll(2)** in **poll\_dispatch**. In this case **poll\_dispatch** simply

returns.

## BUGS

Obscure sequences of **poll\_start\_timer** and **poll\_stop\_timer** in callback functions may probably break the code.

The semantics of **POLL\_EXCEPT** are not clear.

## AUTHORS

Hartmut Brandt, harti@freebsd.org