

Re: pthread help

Source: <http://linux.derkeiler.com/Newsgroups/comp.os.linux.development.apps/2006-11/msg00190.html>

- *From:* "David Schwartz" <davids@xxxxxxxxxxxxxx>
 - *Date:* 10 Nov 2006 19:28:09 -0800
-

gaetanoortisi@xxxxxxxx wrote:

No, suppose I have data stored on the buffer that represent media contents (audio, video). You could have the possibility to rewind, pause, stop, restart, etc. on this contents. The job of the thread is to output these contents until a pause or stop command, in which case it has to (exit, suspend itself , what is better?), and (restart or recreate) in case of a new play command. An approach could be (experimental) a play loop that looks for a flag variable before it starts a cycle of output, and if the variable is negative, do dumb operations. However, about me is not a good solution.

It depends upon your architecture. It's not typical to have the entire media contents sitting in memory.

For example, one architecture might be to have a buffer filling thread that reads data (from a network, from a disk, or computer it) and tries to keep a queue full. Another thread drains that queue by displaying or playing the data in it. A control thread would interact with these two threads the same way they interact with each other.

```
void thread_fun(void)
{
  for (;;) {
    if (flag) {
      .....output .....
    }
  }
}
```

You need to hold a mutex while you test the flag.

```
void run (void)
{
pthread_mutex_lock(&m);
flag=1;
pthread_mutex_unlock(&m);
}

void pause (void)
{
pthread_mutex_lock(&m);
flag=0;
pthread_mutex_unlock(&m);
}
```

Yes, this can work, but it's not a very realistic example. This kind of brute force approach is usually only used to shut down an entire process. Normally, interact with threads in the way they normally determine what work to do, rather than through some special "takeover control" mechanism.

It's kind of like if you have a driver out doing deliveries and suddenly there's a change in the delivery list. It makes more sense just to tell the driver the new places to deliver the same way you were going to tell him the original places he was going to deliver, rather than calling him back to the shop, shooting him, hiring a new driver, and giving him the new list of deliveries.

On the flip side, rewinding or stopping is arguably sufficiently like a shutdown to warrant an "exceptional" control mechanism. But I'd think you'd learn more doing it in a more organized way, simply changing what you were telling the thread to do anyway rather than trying to interrupt it. The way you decide what work to do should be flexible by design and not require a kill switch except for shutdown or error conditions.

DS

.