Making asynchronous system calls in Linux kernel

Student's Individual Study
Sakhybekova Balnur

"Asynchronous I/O" allows for a process to monitor multiple file descriptors and get notifications when I/O is possible on them.

A level-triggered AIO mechanism provides (when queried) information about which AIO events are still pending. In general, this translates to a set of file descriptors on which reading (or writing) can be performed without blocking.

An edge-triggered mechanism on the other hand provides information about which events have changed status (from non-pending to pending) since the last query.

Dealing with asynchronous events on Linux

- Threading
- Signals
- The SIGIO signal
- select() and poll()
- epoll()
- POSIX asynchronous I/O (AIO)

`epoll` is a Linux kernel system call for a scalable I/O event notification mechanism, first introduced in version 2.5.44 of the Linux kernel mainline.[1] Its function is to monitor multiple file descriptors to see whether I/O is possible on any of them. It is meant to replace the older POSIX `select()` and `poll()` system calls, to achieve better performance in more demanding applications, where the number of watched file descriptors is large (unlike the older system calls, which operate in O(n) time, `epoll` operates in O(1) time[2]).

Makefile

```
all: epoll_example

epoll_example: epoll_example.c
  gcc -Wall -Werror -o $@
epoll_example.c

clean:
  @rm -v epoll_example
```

Used libraries

```
#include <stdio.h>      // for fprintf()

#include <unistd.h>     // for close(), read()

#include <sys/epoll.h> // for epoll_create1(), epoll_ctl(), struct epoll_event

#include <string.h>     // for strncmp
```

```c
#include <stdio.h>     // for fprintf()
#include <unistd.h>    // for close()
#include <sys/epoll.h> // for epoll_create1()

int main()
{
  int epoll_fd = epoll_create1(0);

  if(epoll_fd == -1)
  {
    fprintf(stderr, "Failed to create epoll file descriptor\n");
    return 1;
  }

  if(close(epoll_fd))
  {
    fprintf(stderr, "Failed to close epoll file descriptor\n");
    return 1;
  }
  return 0;
}
```

```c
#include <stdio.h>     // for fprintf()
#include <unistd.h>    // for close()
#include <sys/epoll.h> // for epoll_create1(), epoll_ctl(), struct epoll_event

int main()
{
  struct epoll_event event;
  int epoll_fd = epoll_create1(0);

  if(epoll_fd == -1)
  {
    fprintf(stderr, "Failed to create epoll file descriptor\n");
    return 1;
  }

  event.events = EPOLLIN;
  event.data.fd = 0;

  if(epoll_ctl(epoll_fd, EPOLL_CTL_ADD, 0, &event))
  {
    fprintf(stderr, "Failed to add file descriptor to epoll\n");
    close(epoll_fd);
    return 1;
  }

  if(close(epoll_fd))
  {
    fprintf(stderr, "Failed to close epoll file descriptor\n");
    return 1;
  }
  return 0;
}
```

```c
int main()
{
  int running = 1, event_count, i;
  size_t bytes_read;
  char read_buffer[READ_SIZE + 1];
  struct epoll_event event, events[MAX_EVENTS];
  int epoll_fd = epoll_create1(0);

  if(epoll_fd == -1)
  {
    fprintf(stderr, "Failed to create epoll file descriptor\n");
    return 1;
  }

  event.events = EPOLLIN;
  event.data.fd = 0;

  if(epoll_ctl(epoll_fd, EPOLL_CTL_ADD, 0, &event))
  {
    fprintf(stderr, "Failed to add file descriptor to epoll\n");
    close(epoll_fd);
    return 1;
  }

  while(running)
  {
    printf("\nPolling for input...\n");
    event_count = epoll_wait(epoll_fd, events, MAX_EVENTS, 30000);
    printf("%d ready events\n", event_count);
    for(i = 0; i < event_count; i++)
    {
      printf("Reading file descriptor '%d' -- ", events[i].data.fd);
      bytes_read = read(events[i].data.fd, read_buffer, READ_SIZE);
      printf("%zd bytes read.\n", bytes_read);
      read_buffer[bytes_read] = '\0';
      printf("Read '%s'\n", read_buffer);

      if(!strncmp(read_buffer, "stop\n", 5))
        running = 0;
    }
  }

  if(close(epoll_fd))
  {
    fprintf(stderr, "Failed to close epoll file descriptor\n");
    return 1;
  }
  return 0;
}
```

sysprog [Running]

Activities Terminal Cc 18:42

bestriess@bestriess-VirtualBox: ~/sysprog/sis

File Edit View Search Terminal Help

```
epoll_example.c  Makefile
bestriess@bestriess-VirtualBox:~/sysprog/sis$ ./epoll_example
bash: ./epoll_example: No such file or directory
bestriess@bestriess-VirtualBox:~/sysprog/sis$ make
Makefile:4: *** missing separator.  Stop.
bestriess@bestriess-VirtualBox:~/sysprog/sis$ ls
epoll_example.c  Makefile
bestriess@bestriess-VirtualBox:~/sysprog/sis$ gedit Makefile
bestriess@bestriess-VirtualBox:~/sysprog/sis$ make
Makefile:4: *** missing separator.  Stop.
bestriess@bestriess-VirtualBox:~/sysprog/sis$ gcc -o epoll_example.c
gcc: fatal error: no input files
compilation terminated.
bestriess@bestriess-VirtualBox:~/sysprog/sis$ gcc -o epoll_example  epoll_examp
le.c
bestriess@bestriess-VirtualBox:~/sysprog/sis$ ls
epoll_example  epoll_example.c  Makefile
bestriess@bestriess-VirtualBox:~/sysprog/sis$ ./epoll_example

Polling for input...
0 ready events

Polling for input...
```

Left ⌘

sysprog [Running]

Activities Terminal Cc 18:42

bestriess@bestriess-VirtualBox: ~/sysprog/sis

File Edit View Search Terminal Help

```
le.c
bestriess@bestriess-VirtualBox:~/sysprog/sis$ ls
epoll_example  epoll_example.c  Makefile
bestriess@bestriess-VirtualBox:~/sysprog/sis$ ./epoll_example

Polling for input...
0 ready events

Polling for input...

1 ready events
Reading file descriptor '0' -- 1 bytes read.
Read '
'

Polling for input...
hello
1 ready events
Reading file descriptor '0' -- 6 bytes read.
Read 'hello
'

Polling for input...
```

Left ⌘

sysprog [Running]

Activities Terminal Cc 18:43

bestriess@bestriess-VirtualBox: ~/sysprog/sis

File Edit View Search Terminal Help

```
Polling for input...
By Balnur Sakhybekovaaa
1 ready events
Reading file descriptor '0' -- 10 bytes read.
Read 'By Balnur '

Polling for input...
1 ready events
Reading file descriptor '0' -- 10 bytes read.
Read 'Sakhybekov'

Polling for input...
1 ready events
Reading file descriptor '0' -- 4 bytes read.
Read 'aaa
'

Polling for input...
stop
1 ready events
Reading file descriptor '0' -- 5 bytes read.
Read 'stop
'
bestriess@bestriess-VirtualBox:~/sysprog/sis$
```

Left ⌘