I/O Multiplexing

CS 241

April 9, 2014

University of Illinois

Announcements

mp5 due today

mp6 released today

- MapReduce

Review: Interprocess communication

Shared address space

- Shared memory
- Memory mapped files

Via OS

- Files
- Pipes
- FIFOs (named pipes): Review today
- Signals: New today


SurveyMonkey

Review: FIFOs and dup()

How could we read from a FIFO as if it were stdin?

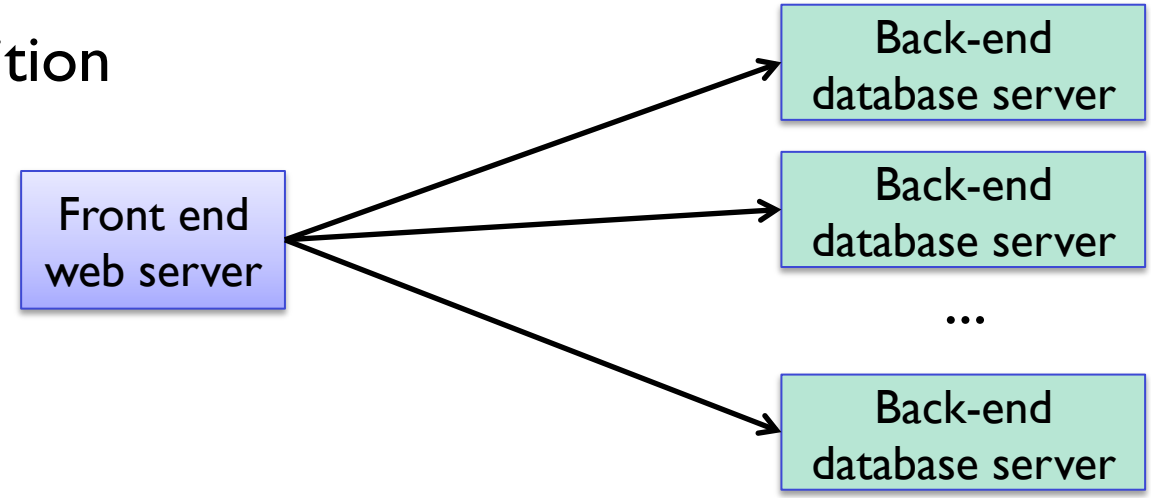
```
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>

int main(int argc, char** argv) {
    mkfifo(argv[1], S_IRWXU | S_IRWXG | S_IRWXO);

    int fifo = open(argv[1], O_RDONLY);

    dup2(fifo, 0); /* 0 is the file descriptor of stdin */

    char line[1024];
    while (fgets(line, 1024, stdin))
        printf("I got this: %s\n", line);
}
```

pipestdin.c

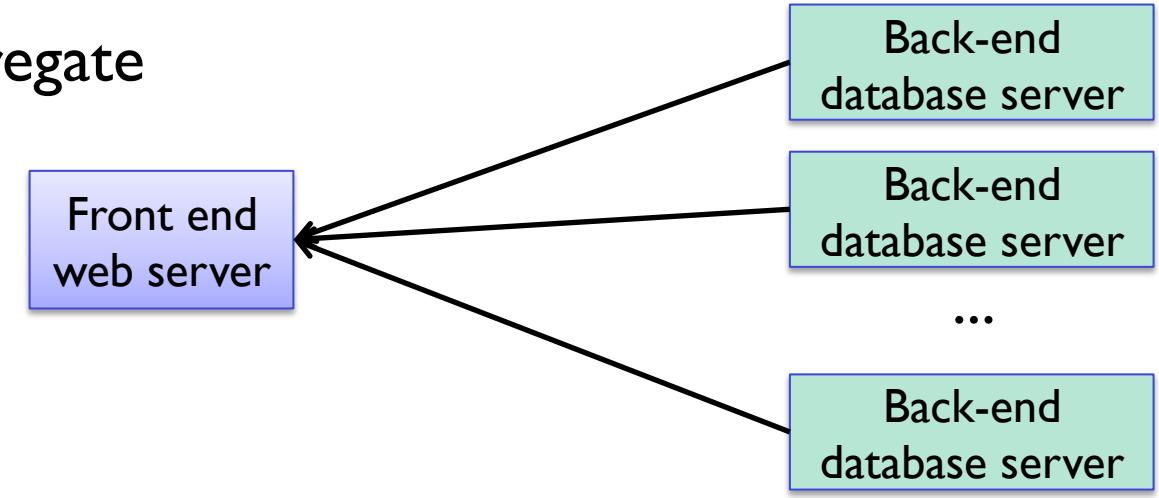
I/O Multiplexing: epoll

Partition/aggregate pattern

1. Partition



2. Aggregate



I/O Multiplexing

By default: `read()` / `fread()` are blocking calls.

- …if no data is available, the process will be moved to the BLOCKED state until data is available.

In order to `read()` from multiple files in one thread at one time, I/O multiplexing is required.

- **`epoll()`:** monitor multiple file descriptors, waiting until one or more of the file descriptors become "ready".

epoll() Overview

Usage of epoll():

- Create an epoll instance via `epoll_create()`
- Register each file descriptor to watch via `epoll_ctl()`
- Use `epoll_wait()` to block until an fd is ready

On Linux, epoll replaces both `select()` and `poll()`

epoll() Overview

epoll_ctl():

```
int epoll_ctl(int epfd, int op, int fd,
              struct epoll_event *event);

op: EPOLL_CTL_ADD: Add to the epoll set
    EPOLL_CTL_MOD: Modify the epoll set
    EPOLL_CTL_DEL: Delete from the epoll set

event:
  struct epoll_event {
     uint32_t     events;      /* Epoll events */
     epoll_data_t data;        /* User data */
  };

  typedef union epoll_data {
     int fd;
     ...   // ...other stuff we will not use.
  } epoll_data_t;
```

epoll() Example

	Process 1	***Process 2***
0s:	*A*	
1s:		*B*
2s:		*C*
3s:	*D*	

epoll() Example (switch to code...)