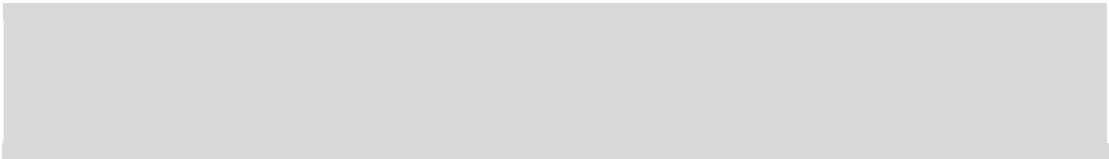


[PDZ99]	GUI	[O96]	
concurrency		node.js[N13]	event-based C/UNIX



%/a#

event-based concurrency

I/O

event loop

```
while (1) {
  events = getEvents();
  for (e in events)
    processEvent(e);
}
```

getEvents()
event handler

I/O

3B; eWUf/i ba7fi

API select() poll()

I/O

Web

select() macOS X API

```
int select(int nfd,
           fd_set *restrict readfds,
           fd_set *restrict writefds,
           fd_set *restrict errorfds,
           struct timeval *restrict timeout);
```

errorfds select() I/O readfds writefds

nfd 0 nfd-1

select()

select()

synchronous

asynchronous

I/O

I/O

select()

NULL select()

select()

poll()

Stevens Rago

[SR05]

%%%

eWWWf/i

select()

33.1

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <sys/time.h>
4  #include <sys/types.h>
5  #include <unistd.h>
6
7  int main(void) {
8      // open and set up a bunch of sockets (not shown)
9      // main loop
10     while (1) {
11         // initialize the fd_set to all zero
12         fd_set readFDs;
13         FD_ZERO(&readFDs);
14
15         // now set the bits for the descriptors
16         // this server is interested in
17         // (for simplicity, all of them from min to max)
18         int fd;
19         for (fd = minFD; fd < maxFD; fd++)
20             FD_SET(fd, &readFDs);
21
22         // do the select
23         int rc = select(maxFD+1, &readFDs, NULL, NULL, NULL);
24
25         // check which actually have data using FD_ISSET()
26         int fd;
27         for (fd = minFD; fd < maxFD; fd++)
28             if (FD_ISSET(fd, &readFDs))
29                 processFD(fd);
30     }
31 }
```

33.1 select()

FD_ZERO()

FD_SET()

minFD

maxFD

select()

FD_ISSET()

I/O

Stevens

Rago

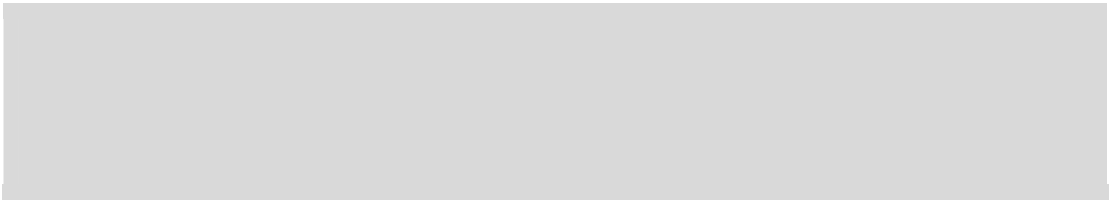
[SR05]

API

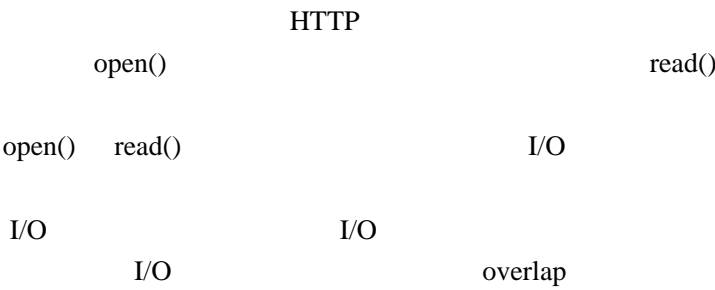
Pai Welsh [PDZ99 WCB01]

%~&
%~&

CPU



%~&



%~&

;! A

I/O

I/O asynchronous I/O

I/O

I/O

macOS X

I/O

API

struct aiocb

AIO

AIO control block

API

```

struct aiocb {
    int          aio_fildes;    /* File descriptor */
    off_t        aio_offset;    /* File offset */
    volatile void *aio_buf;     /* Location of buffer */
    size_t       aio_nbytes;    /* Length of transfer */
};

```

aio_fildes

ai_offset

aio_nbytes

aio_buf

macOS X

API

asynchronous read API

```
int aio_read(struct aiocb *aiocbp);
```

I/O

I/O

aio buf

API

macOS X

aio_error()

API

```
int aio_error(const struct aiocb *aiocbp);
```

aiocbp

EINPROGRESS

I/O

aio_error()

poll

I/O

I/O

I/O

interrupt

UNIX

signal

I/O

I/O

UNIX

UNIX

signal

signal handler

HUP INT SEGV

SIGSEGV SIG

SEGV

SIGHUP

```
#include <stdio.h>
#include <signal.h>

void handle(int arg) {
    printf("stop wakin' me up...\n");
}

int main(int argc, char *argv[]) {
    signal(SIGHUP, handle);
    while (1)
        ; // doin' nothin' except catchin' some sigs
    return 0;
}
```

kill handle()

while

```
prompt> ./main &
[3] 36705
prompt> kill -HUP 36705
stop wakin' me up...
prompt> kill -HUP 36705
stop wakin' me up...
prompt> kill -HUP 36705
stop wakin' me up...
```

Stevens Rago [SR05]

I/O

Pai [PDZ99]

I/O

%/2)

I/O

I/O

Adya

manual stack management

[A + 02]

fd

SD

```
int rc = read(fd, buffer, size);  
rc = write(sd, buffer, size);
```

read()

sd

AIO

aio_error()

Adya

[A+02]

continuation

[FHK84]

I/O

sd

fd

I/O

%/%*

CPU

CPU

CPU

paging

[A+02]

API

I/O
 [PDZ99] I/O
 select() I/O select()
 I/O AIO

%/z+

[A+02 PDZ99
 vB+03 WCB01]

[A+02] Cooperative Task Management Without Manual Stack Management Atul Adya, Jon Howell, Marvin Theimer, William J. Bolosky, John R. Douceur USENIX ATC 92, Monterey, CA, June 2002

[FHK84] Programming With Continuations
 Daniel P. Friedman, Christopher T. Haynes, Eugene E. Kohlbecker
 In Program Transformation and Programming Environments, Springer Verlag, 1984

[N13] Node.js Documentation By the folks who build node.js
 Web

[O96] Why Threads Are A Bad Idea (for most purposes) John Ousterhout
 Invited Talk at USENIX 96, San Diego, CA, January 1996

GUI Ousterhout
 Tcl/Tk Tcl/Tk
 GUI 100 Tk GUI Python Tcl

[PDZ99] Flash: An Efficient and Portable Web Server Vivek S. Pai, Peter Druschel, Willy Zwaenepoel
 USENIX 99, Monterey, CA, June 1999

Web

I/O

[SR05] Advanced Programming in the UNIX Environment

W. Richard Stevens and Stephen A. Rago Addison-Wesley, 2005

UNIX

[vB+03] Capriccio: Scalable Threads for Internet Services

Rob von Behren, Jeremy Condit, Feng Zhou, George C. Necula, Eric Brewer SOSP 03, Lake George, New York, October 2003

[WCB01] SEDA: An Architecture for Well-Conditioned, Scalable Internet Services Matt Welsh, David Culler, and Eric Brewer

SOSP 01, Banff, Canada, October 2001