Humboldt University

Computer Science Department
Systems Architecture Group
http://sar.informatik.hu-berlin.de

Operating Systems Principles

Event Queue

Events

Aufgabe

- 2 Wochen Zeit zum Lösen der Aufgaben
  - Aufgabenstellung auf der SAR Website
  - Abgabe über GOYA
  - Abgabefrist: 17.11.2013 09:00
- In dieser Woche
  - Erläutern der Aufgabenstellung
  - Nächste Woche: Zeit zur Bearbeitung
- Nächste Veranstaltung
  - 14.11.2013

Aufgabe

Aufgabe

Implementieren Sie eine Event Queue (EQ) auf Grundlage der vorgegeben API.

Folgende Arten von Events sollen behandelt werden können:

1. Tastatureingaben
2. UDP-Sockets (Empfang)
3. Named Pipes
4. Timer/Timeouts

Für jedes dieser Events können mehrere Callbacks (Funktionen) in die Queue eingefügt werden. Beim Auftreten eines Events (z.B. Tastutur-eingabe) sollen die entsprechenden Funktionen aufgerufen werden.

Timer events werden nur einmalig aufgerufen. Bei ihnen gibt man die Zeit an,wann die Funktion aufgerufen werden soll.

Details

Vorgabe

Folgende Vorgaben gibt es:

- Makefile
- event.(h|c)
- event_queue.(h|c)
- eventqueuetest.c

Die Datei event.h enthält die 4 Arten von Events:

- SOCKET
- PIPE
- KEY
- TIMEOUT

Systems Architecture Group
http://sar.informatik.hu-berlin.de

Events

- Programmieren mit Events unter Unix:
  - Signale
  - *select()*
  - *libboost (asio)*

- Event Queues (z.B in Simulatoren)
  - Jist
  - NS2

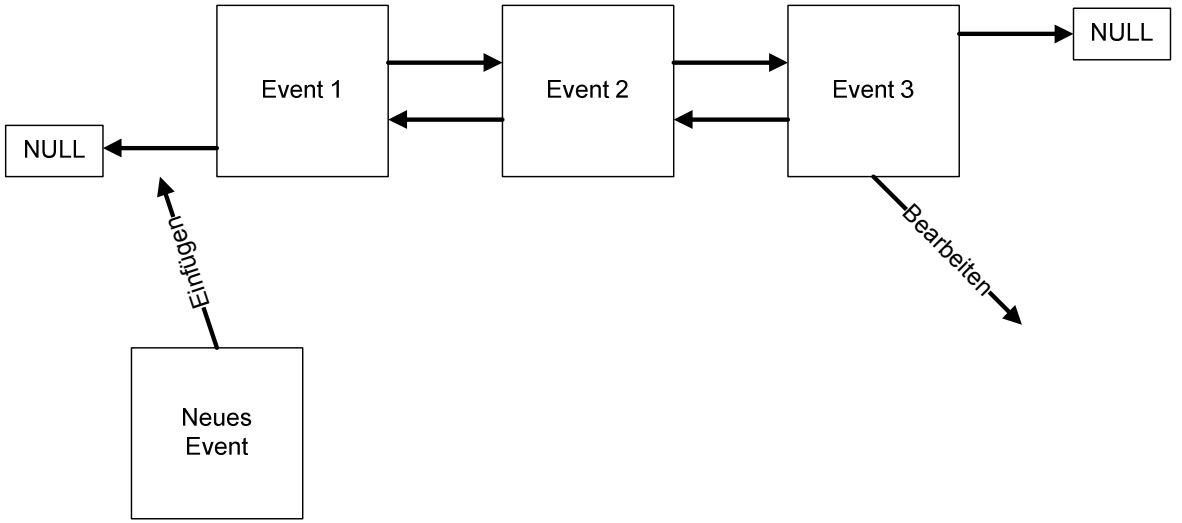
Systems Architecture Group
http://sar.informatik.hu-berlin.de

Eventqueue

Eventqueue

- Event
  - Priorität
  - *Handler*
  - *Zeit*
  - *…*

- Event Queue
  - Verkettete Liste
  - Events:
    - Löschen
    - Einfügen
  - Scheduler arbeitet Events ab



Systems Architecture Group
http://sar.informatik.hu-berlin.de

Signale

signal

NAME

signal

SYNOPSIS

#include <signal.h>

void (*signal(int *sig*, void (**func*)(int)))(int);

DESCRIPTION

signal management

Determines how to handle the receiption of the signal number *sig* is to be subsequently handled

- SIG_DFL - default handling for that signal
- SIG_IGN - the signal shall be ignored
- Otherwise, *func* points to a function ("signal handler") to be called when that signal occurs

http://www.opengroup.org/onlinepubs/009695399/functions/signal.html

wait

NAME

wait, waitpid

SYNOPSIS

#include <sys/wait.h>

pid_t wait(int *stat_loc);
pid_t waitpid(pid_t *pid*, int *stat_loc*, int *options*);

DESCRIPTION

Wait for a child process to stop or terminate

Obtain status information pertaining to one of the caller's child processes

If status information is available for two or more child processes, the order in which their status is reported is unspecified.

http://www.opengroup.org/onlinepubs/009695399/functions/wait.html

kill

NAME

kill

SYNOPSIS

#include <signal.h>

int kill(pid_t pid, int sig);

DESCRIPTION

Sends signal to a process or a group of processes specified by *pid*

Pids <= 0 used to send a signal to more than one process

Return value: 0 if successful ; -1 otherwise

Signals (examples):

SIGSTOP, SIGCONT

SIGTERM, SIGKILL

http://pubs.opengroup.org/onlinepubs/7908799/xsh/kill.html

Select

Systems Architecture Group
http://sar.informatik.hu-berlin.de

select

NAME

select

SYNOPSIS

#include <sys/select.h>

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

int FD_ISSET(int *fd*, fd_set **fdset*);
void FD_SET(int *fd*, fd_set **fdset*);
void FD_ZERO(fd_set **fdset*);

DESCRIPTION

select() return if one or more of the file descriptor in the sets (*readfds*, *writefds*, and *errorfds)* are ready for reading, are ready for writing, or have an exceptional condition pending or due to a timeout

struct timeval *restrict timeout

- gibt Dauer bis zum Timeout an
- Enthält die Restzeit bis zum Timeout, wenn Daten von einem Filedescriptor abgeholt werden können

```
tv.tv_sec = QUEUE_TIMEOUT;
tv.tv_usec = 0;
while (1) {
        FD_ZERO(&rfds);
        FD_SET(eq->fd_keystroke_event, &rfds);
        FD_SET(eq->fd_namedpipe_event, &rfds);
        FD_SET(eq->fd_socket_event, &rfds);
        retval = select(max_fd+1, &rfds, NULL, NULL, &tv);
        …
        if (timeout) {
                tv.tv_sec = QUEUE_TIMEOUT;
                tv.tv_usec = 0;
        }
}
```

mkfifo

NAME

mkfifo

SYNOPSIS

```
#include <sys/stat.h>
int mkfifo(const char *path, mode_t mode);
```

DESCRIPTION

The *mkfifo*() function shall create a new FIFO special file named by the pathname pointed to by *path*. The file permission bits of the new FIFO shall be initialized from *mode*. The file permission bits of the *mode* argument shall be modified by the process' file creation mask.

unlink

NAME

unlink - remove a directory entry

SYNOPSIS

```
#include <unistd.h>
int unlink(const char *path);
```

DESCRIPTION

The *unlink*() function shall remove a link to a file. If *path* names a symbolic link, *unlink*() shall remove the symbolic link named by *path* and shall not affect any file or directory named by the contents of the symbolic link. Otherwise, *unlink*() shall remove the link named by the pathname pointed to by *path* and shall decrement the link count of the file referenced by the link.

read/write

NAME

read - read from a file

write - write on a file

SYNOPSIS

#include <unistd.h>

ssize_t read(int *fildes*, void **buf*, size_t *nbyte*);

ssize_t write(int *fildes*, const void **buf*, size_t *nbyte*);

DESCRIPTION

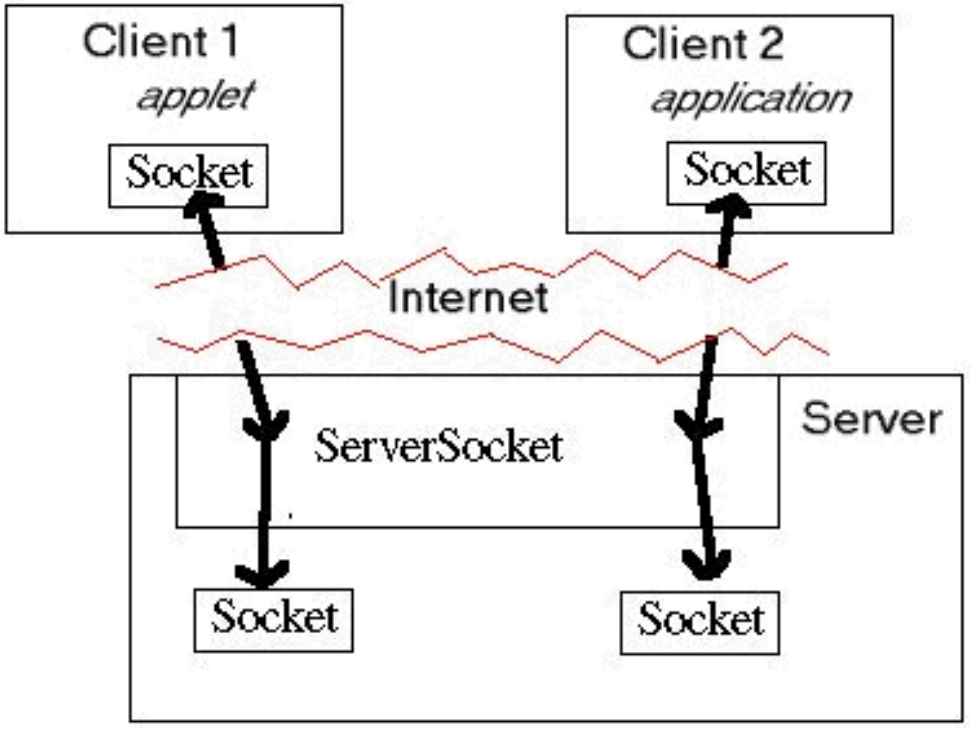
The *read*() function shall attempt to read *nbyte* bytes from the file associated with the open file descriptor. The *write*() function shall attempt to write *nbyte* bytes from the buffer pointed to by *buf* to the file associated with the open file descriptor, *fildes*.

Sockets

Socket

is one endpoint of a two-way communication link between two programs running on the network

is bound to a port number so that the TCP layer can identify the application that data is destined to be sent

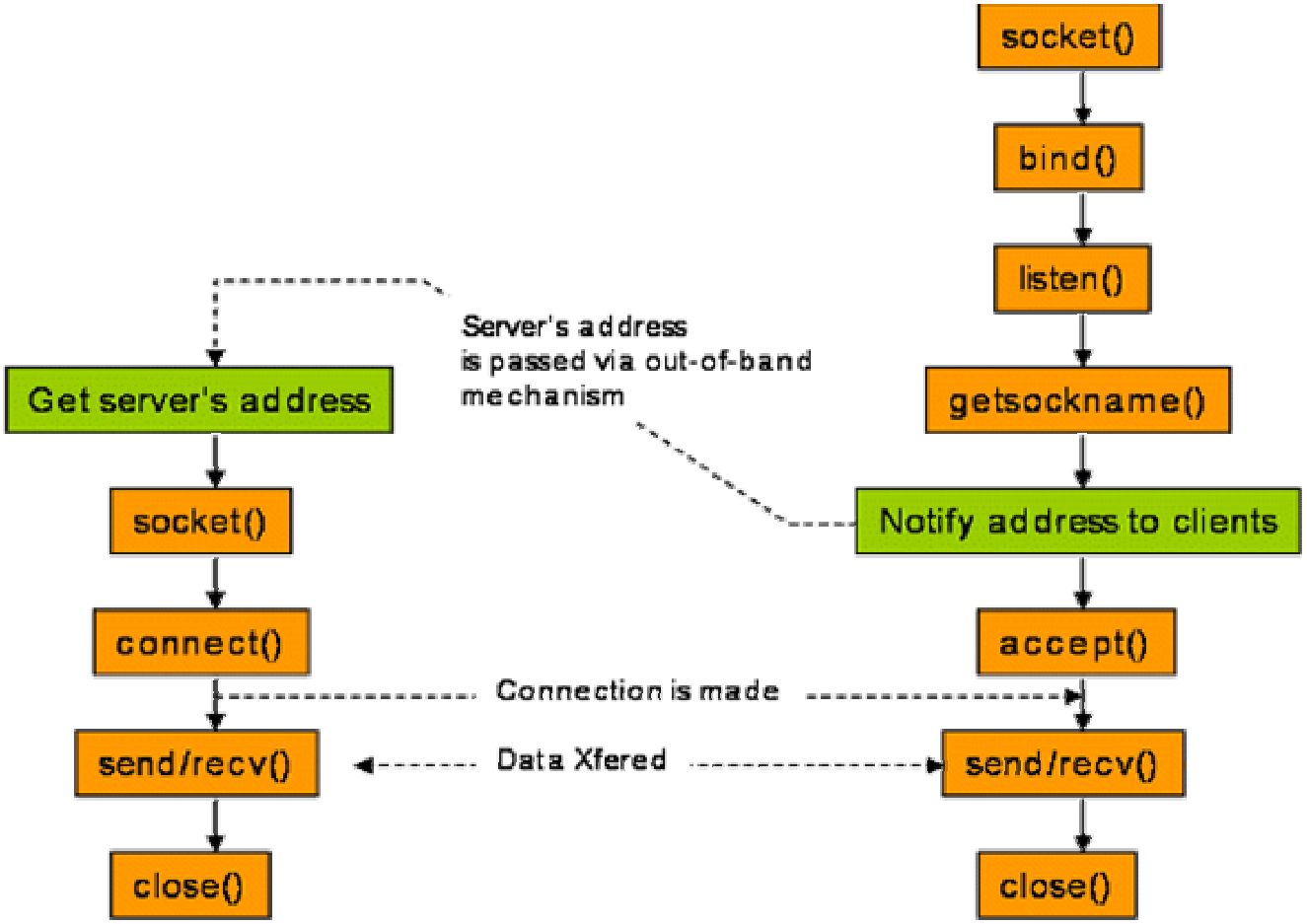


Systems Architecture Group
http://sar.informatik.hu-berlin.de

Sockets

Socket-API

socket, bind, connect,…..



Systems Architecture Group
http://sar.informatik.hu-berlin.de

Sockets

```
struct sockaddr {
   unsigned short    sa_family;    // address family, AF_xxx
   char              sa_data[14];  // 14 bytes of protocol address
};
```

IPv4 sockets (AF_INET)

```
struct sockaddr_in {
    short          sin_family;   // e.g. AF_INET, AF_INET6
    unsigned short   sin_port;     // e.g. htons(3490)
    struct in_addr   sin_addr;     // see struct in_addr, below
    char           sin_zero[8];  // zero this if you want to
};
```

Sockets

#include <sys/socket.h>
int socket(int domain, int type, int protocol);

create an unbound socket in a communications domain
Return value: file descriptor

domain
Specifies the communications domain in which a socket is to be created
(e.g. AF_INET)

type
Specifies the type of socket to be created.
SOCK_STREAM -> TCP
SOCK_DGRAM -> UDP

protocol
Specifies a particular protocol to be used with the socket.

Socket

#include <sys/socket.h>

int bind(int socket, const struct sockaddr *address, socklen_t address_len);

assign a local socket address address to a socket identified by descriptor socket

socket

Specifies the file descriptor of the socket to be bound.

address

Points to a sockaddr structure containing the address to be bound to the socket. The length and format of the address depend on the address family of the socket.

address_len

Length of the address

Socket

```
#include <sys/socket.h>
int listen(int socket, int backlog);
```

listen

listen for socket connections

backlog

Limit the queue of incoming connections

Systems Architecture Group
http://sar.informatik.hu-berlin.de

Socket

```
#include <sys/types.h>
#include <sys/socket.h>
int connect(int sockfd, const struct sockaddr *addr, socklen_t
    addrlen);
```

connects the socket referred to by the file descriptor *sockfd*

to the address specified by *addr*

addrlen argument specifies the size of *addr*.

Socket

```
#include <sys/types.h>
#include <sys/socket.h>

int accept(int sockfd, struct sockaddr *addr, socklen_t
    *addrlen);
```

extracts the first connection request on the queue of pending connections for the listening socket

recvfrom

NAME

recvfrom - receive a message from a socket

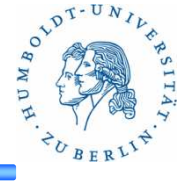
SYNOPSIS

#include <sys/socket.h>

ssize_t recvfrom(int *socket*, void **buffer*, size_t *length*, int *flags*, struct sockaddr **address*, socklen_t **address_len*);

DESCRIPTION

The *recvfrom()* function receives a message from a connection-mode or connectionless-mode socket. It is normally used with connectionless-mode sockets because it permits the application to retrieve the source address of received data.



Threads

POSIX-Threads (Pthreads)

Standard IEEE P1003.1c

In vielen Systemen realisiert

Konzepte in anderen Thread-Realisierungen meist ähnlich

POSIX-konforme Realisierungen unter Linux

Systems Architecture Group
http://sar.informatik.hu-berlin.de

Die Pthread-Schnittstelle

Thread-Erzeugung

Programmiermodell

- Bei Start eines Prozesses existiert genau ein Thread
- Dieser erzeugt ggf. weitere Threads und wartet auf deren Ende
- Terminierung des Prozesses bei Terminierung des Master-Threads

Funktion zur Thread-Erzeugung

```
int pthread_create(
        pthread_t *new_thrd_ID,
        const pthread_attr_t *attr,
        void *(*start_func)(void *),
        void *arg)
```


Systems Architecture Group
http://sar.informatik.hu-berlin.de

Die Pthread-Schnittstelle

Thread-Attribute

Stackgröße

Scheduling-Schema und Priorität

detachstate: Verhalten bei Beendigung desThreads

Bei **detached thread** erfolgt die Freigabe der Ressourcen sofort bei Beendigung, ansonsten erst nach Abschlußsynchronisation

Die Pthread-Schnittstelle

Thread-Verwaltung

pthread_self

Liefert eigene Thread-ID

pthread_exit

Eigene Terminierung

pthread_cancel

Terminiert einen Thread

Terminierung nur an bestimmten Punkten

Vor Terminierung **cleanup handler** aufrufen

Die Pthread-Schnittstelle

Thread-Verwaltung...

pthread_join

Wartet auf Terminierung des spezifizierten Threads (Abschlußsynchronisation)

pthread_sigmask

Setzt Signalmaske (jeder Thread hat eigene Maske)

pthread_kill

Sendet Signal an anderen Thread innerhalb des Prozesses

Von außen nur Signal an irgendeinen Thread möglich

Systems Architecture Group
http://sar.informatik.hu-berlin.de

Die Pthread-Schnittstelle

Mutex-Operation (wechselseitiger Ausschluß)

pthread_mutex_init

Initialisiert Sperrvariable (mutex)

pthread_mutex_destroy

pthread_mutex_lock

Blockiert Thread, bis Sperre frei ist und belegt dann die Sperre

pthread_mutex_trylock

Belegt Sperre, falls möglich / kein Blockieren

pthread_mutex_unlock

Anmerkungen

Bei Terminierung eines Threads werden Sperren nicht automatisch freigegeben