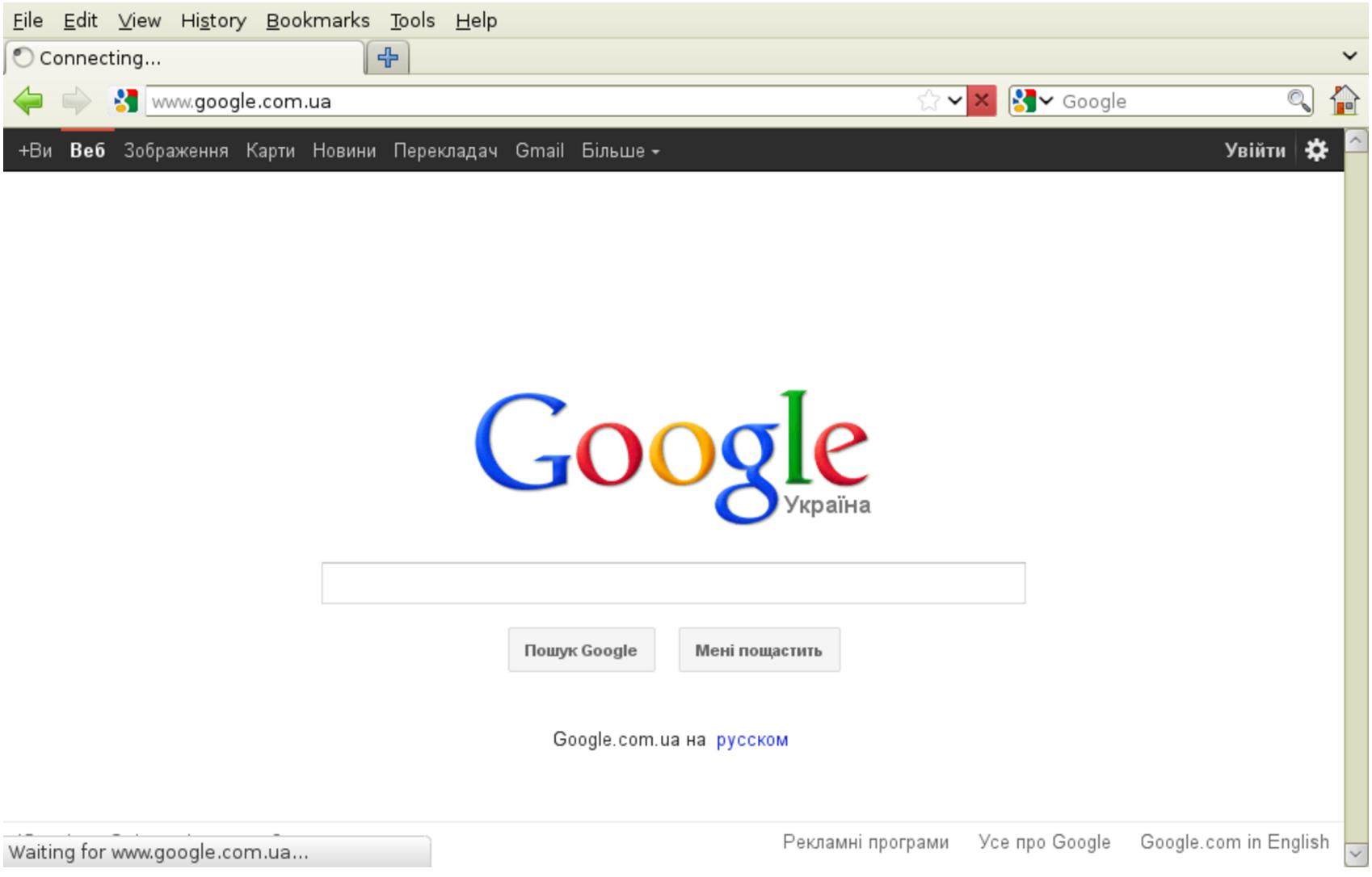
Мультиплексирование ввода-вывода

Роман Чепляка

29 ноября 2011
НТУУ КПИ

Пример: браузер



read

```
ssize_t read(int fd, void *buf, size_t count);
```

Потоки

You have a problem and decide to use threads.

two Now problems you have.

Потоки

Потоки:

+ Простота

Потоки

Потоки:

- \+ Простота
- \+ Модульность

Потоки

Потоки:

- \+ Простота
- \+ Модульность
- – Потребление памяти

Потоки

Потоки:

- \+ Простота
- \+ Модульность
- – Потребление памяти
- – Переключения контекста

Неблокирующий ввод-вывод

```
#include <fcntl.h>

for ( i = 0; i < n; i ++ ) {
    fcntl(socket[i], F_SETFD, O_NONBLOCK);
}

while (1) {
    for ( i = 0; i < n; i ++ ) {
        if ( (r = read(socket[i], buf, BUFSIZE)) >= 0 )
            { ... }
    }
}
```

Are we there yet?

Блокирующий или неблокирующий?

Блокирующий режим	**Неблокирующий режим**
Данные всегда есть	Данных может не быть
Одно событие	Много событий
push-модель	pull-модель

epoll

Epoll:

- Linux-only
- Стандартизированные, но менее эффективные аналоги: select, poll
- Эффективные аналоги для других систем: /dev/poll, kqueue, ...

epoll

1. Регистрируем интересные файловые дескрипторы
2. Вызываем

   `epoll_wait`
3. Блокируемся, пока не появятся данные в одном из файловых дескрипторов

epoll

```
while (1) {
    int nfds = epoll_wait(epollfd, events, MAX_EVENTS, -1);

    for (n = 0; n < nfds; n++) {
        int fd = events[n].data.fd;
        while ( (r = read(fd, buf, BUFSIZE)) > 0 )
            { ... }
    }
}
```

epoll

Epoll:

- push-модель
- один поток
- не кросс-платформенный

Низкий уровень

- Потоки
- Средства мультиплексирования: epoll

libevent

libevent: решение проблемы кросс-платформенности
Аналог: libev

Пример

```c
int n;

printf("Enter the number: ");
scanf("%d", &n);

printf("You entered %d\n", n);
```

Пример

```
int main() {
    struct event ev;
    event_init();
    event_set(&ev, 0, EV_READ, callback, NULL);
    event_add(&ev, NULL);

    printf("Please enter a number\n");

    event_dispatch();
}
```

Пример

```
void callback(int fd, short event, void *arg) {
    int n; FILE *f;
    f = fdopen(fd, "r");

    fscanf(f, "%d", &n);

    printf("You entered %d\n", n);
}
```

Пример

```c
int n1, n2;

printf("Enter the first number: ");
scanf("%d", &n1);

printf("Enter the second number: ");
scanf("%d", &n2);

printf("The sum of %d and %d is %d\n", n1, n2, n1+n2);
```

Пример

```
int main() {
    struct event ev;
    event_init();
    event_set(&ev, 0, EV_READ, callback1, NULL);
    event_add(&ev, NULL);
    printf("Enter the first number:\n");
    event_dispatch();
}
```

Пример

```
void callback1(int fd, short event, void *arg) {
    int *n;
    FILE *f;
    struct event *ev;
    n = malloc(sizeof(int));
    ev = malloc(sizeof(struct event));
    f = fdopen(dup(fd), "r");
    fscanf(f, "%d", n);
    fclose(f);

    printf("Enter the second number:\n");

    event_set(ev, 0, EV_READ, callback2, n);
    event_add(ev, NULL);
}
```

Пример

```
void callback2(int fd, short event, void *arg) {
    int n1, n2;
    FILE *f;
    n1 = *((int*)arg);
    f = fdopen(dup(fd), "r");
    fscanf(f, "%d", &n2);
    fclose(f);

    printf("The sum of %d and %d is %d\n", n1, n2, n1+n2);
}
```

node.js

```
process.stdout.write("Enter the first number\n");
process.stdin.once('data', function (n1) {
    process.stdout.write("Enter the second number\n");
    process.stdin.once('data', function (n2) {
        process.stdout.write(
            (parseInt(n1)+parseInt(n2)).toString());
         })
    });
```

Continuation-passing C

Автоматическое CPS-преобразование

Средний уровень

- libevent
- node.js
- Continuation-passing C

Кооперативная многозадачность

Надо быть осторожным, чтобы не заблокировать единственный поток

Высокий уровень

- Легковесные потоки: Haskell, Erlang
- Ввод-вывод выполняется в отдельном потоке с помощью epoll
- Вычислительные потоки исполняются в фиксированном числе ОС-потоков
- Вытесняющая многозадачность

Пишите письма

roma@ro-che.info