

لينكس في `poll` استخدام

جاهزة ال بيانات تكون عندما مثل ملفات، معالمة عدة على فعال بشكل ال فعاليات لتتبع مصممة لينكس نظام مكاملة هو `epoll` معالمة إلى ال خادم يحتاج قد حيث الشبكية، برمجة في خاص بشكل مفيد فهو للكتابة. جاهز المرفق يكون عندما أو للقراءة ك بير عدد مع أفضل بشكل يتسق `epoll` فإن، `poll` أو `select` مثل قديمة بأجهزة مقارنة متزامن. بشكل الإتصالات من ال عديد للاستعداد. متكرر بشكل ال معالمة جميع فحصي تجنب لأنه ال ملفات معالمة من

ال بيانات. وتعيد الإتصالات تقبل لخدمة بسيطة بامثلة متبوعاً لينكس، في `epoll` لاستخدام بخطوة خطوة دليل هنا

`epoll` لاستخدام خطوات

1. `epoll` مثيل إنشاء

`epoll_create1` ال نظام مكاملة استخدم ملف. معلّم يمثله والذي، `epoll` مثيل إنشاء إلى تحتاج، `epoll` استخدام لبدء

```
int epoll_fd = epoll_create1(0);
```

لأغراض `EPOLL_CLOEXEC` مثل علامات استخدام يمكن خاصة. علامات بدون الأساسي لل استخدام 0 بتقديم قم: ال معلمة مقدمة.

على للحصول `errno` من تحقق الخطأ عند -1 أو النجاح، عند `epoll_fd` ملف معلّم ترجع: تعود التي القمية ال تفاصيل.

المفضل. هو `epoll_create1` فإن لذا الآن، تتجاهل الحجم نصيحة تأخذ ولكن مشابهة `epoll_create` القديمة المكاملة

2. للتابع ملفات معالمة إضافة

تتبعها: تريده التي ال فعاليات وتحديده `epoll` مثيل مع المرفقات `epoll` مثل ال ملفات معالمة لتسجيل `epoll_ctl` استخدم

```
struct epoll_event ev;
ev.events = EPOLLIN; //
ev.data.fd = some_fd; //
epoll_ctl(epoll_fd, EPOLL_CTL_ADD, some_fd, &ev);
```

ال معلّمات:

- `epoll_fd`: `epoll` مثيل معلّم.
- `EPOLL_CTL_ADD`: معلّم لإضافة عمليّة.
- `some_fd`: مرفق. مثل تتبعه سيتم الذي ال ملف معلّم.
- `&ev`: ال إضافية وال معلّمات ال فعاليات يحدد `struct epoll_event` إلى مؤشر.

ال شرائعة ال فعاليات:

- `EPOLLIN`: جاهزة البيانات للقراءة.
- `EPOLLOUT`: جاهز للكتابة.
- `EPOLLERR`: خطأ حدث.
- `EPOLLHUP`: إغلاق مثل الاتصال قطع.

□ مثل أخرى بيانات أو موضح □ هو □ كما ملف مع `struct epoll_event` في `data` حقل تخزن أن يمكن: المستخدم بيانات □
 □ الفعاليات. حدوث عند المصدر لتحديد مؤشر □

3. الفعاليات انتظار

تتبعه: يتم التي الملفات مع الم على الفعاليات وانتظار لتوقف `epoll_wait` استخدم

```
struct epoll_event events[MAX_EVENTS];
int nfd = epoll_wait(epoll_fd, events, MAX_EVENTS, -1);
```

□ المعلمات:

- `epoll_fd`: `epoll` مثيل.
- `events`: الفعاليات لتخزين مصفوفة.
- `MAX_EVENTS`: الحد الأقصى للفعاليات عدد.
- `-1`: فوراً. يعود 0 نهاية؛ بل انتظار يعني `-1` ثانية بالملي الانتظار وقت: `-1`.

□ الخطأ. عند `-1` أو `nfd` الفعاليات على تحتوي التي الملفات مع الم عدد: تعود التي القيمة □

4. الفعاليات معالجة

وعمليته: `epoll_wait` بواسطة إرجاعه تم التي الفعاليات على عبر

```
for (int i = 0; i < nfd; i++) {
    if (events[i].events & EPOLLIN) {
        // `events[i].data.fd`
    }
}
```

□ الفعالية. نوع لتحديد `events[i].events & EPOLLIN` مثل المنطقية العمليات باستخدام `events` حقل من تحقق □
 □ الفعالية. في تسبب الذي الملف مع الم لتحديد `events[i].data.fd` استخدم □

5. اختياري الملفات مع الم إدارة

□ ملف: مع الم تتبع لإيقاف `EPOLL_CTL_DEL` مع `epoll_ctl` استخدم: إزالة □

```
epoll_ctl(epoll_fd, EPOLL_CTL_DEL, some_fd, NULL);
```

□ `EPOLL_CTL_MOD` مع الفعاليات تعديل: تعديل □

```
ev.events = EPOLLOUT; //
epoll_ctl(epoll_fd, EPOLL_CTL_MOD, some_fd, &ev);
```

رئيسية مفاهيم

حافة على تتبع مقابل مستوى على التتبع

- المقروءة: غير البيانات مثل الوضع استمر طالما متكرر بشكل epoll يبلغ: الافتراضي مستوى على التتبع الحلالات. معظم في أسهل
- يطلب جديدة: بيانات وصول مثل الوضع يتغير عندما واحدة مرة فقط يبلغ: EPOLLET حافة على التتبع تعقيداً. أكثر ولكن لفاءة أكثر الفعاليات؛ فقدان لمنع EAGAIN حتى البيانات جميع قراءة كتابة
- الحافة: تتبع وضع تستخدم كنت إذا EPOLLIN | EPOLLET مثل ev.events في EPOLLET بتحديث م

متزامن غير إدخال إخراج

وضع إلى مرفق بتحديث م الإدخال الإخراج. عمليات على التوقف لمنع متزامن غير ملفات معالماً مع غالباً epoll يستخدم باسخدام: متزامن غير

```
fcntl(fd, F_SETFL, fcntl(fd, F_GETFL) | O_NONBLOCK);
```

بسيط إكواد مثال:

من المستوى تتبع وضع يستخدم العملء، إلى البيانات وإعادة الاتصالات لتقبي epoll تستخدم لخدمة أساسية مثال هنا السهولة. أجل

```
#include <sys/epoll.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>

#define MAX_EVENTS 10
#define PORT 8080
```

```

int main() {
    //
    int listen_fd = socket(AF_INET, SOCK_STREAM, 0);
    if (listen_fd == -1) { perror("socket"); exit(1); }

    struct sockaddr_in addr = { .sin_family = AF_INET, .sin_addr.s_addr = INADDR_ANY, .sin_port = htons(PORT)
    if (bind(listen_fd, (struct sockaddr*)&addr, sizeof(addr)) == -1) { perror("bind"); exit(1); }
    if (listen(listen_fd, 5) == -1) { perror("listen"); exit(1); }

    //
    fcntl(listen_fd, F_SETFL, fcntl(listen_fd, F_GETFL) | O_NONBLOCK);

    //      `epoll`
    int epoll_fd = epoll_create1(0);
    if (epoll_fd == -1) { perror("epoll_create1"); exit(1); }

    //      `epoll`
    struct epoll_event ev, events[MAX_EVENTS];
    ev.events = EPOLLIN; //
    ev.data.fd = listen_fd;
    if (epoll_ctl(epoll_fd, EPOLL_CTL_ADD, listen_fd, &ev) == -1) { perror("epoll_ctl"); exit(1); }

    //
    while (1) {
        int nfds = epoll_wait(epoll_fd, events, MAX_EVENTS, -1);
        if (nfds == -1) { perror("epoll_wait"); exit(1); }

        for (int i = 0; i < nfds; i++) {
            int fd = events[i].data.fd;

            if (fd == listen_fd) {
                //
                int client_fd = accept(listen_fd, NULL, NULL);
                if (client_fd == -1) { perror("accept"); continue; }

                //
                fcntl(client_fd, F_SETFL, fcntl(client_fd, F_GETFL) | O_NONBLOCK);

                //      `epoll`
                ev.events = EPOLLIN;

```

```

    ev.data.fd = client_fd;
    if (epoll_ctl(epoll_fd, EPOLL_CTL_ADD, client_fd, &ev) == -1) {
        perror("epoll_ctl: client_fd");
        close(client_fd);
    }
} else {
    //
    char buf[1024];
    ssize_t n = read(fd, buf, sizeof(buf));
    if (n <= 0) {
        //      :      `epoll`
        if (n == -1 && (errno == EAGAIN || errno == EWOULDBLOCK)) continue;
        close(fd);
        epoll_ctl(epoll_fd, EPOLL_CTL_DEL, fd, NULL);
    } else {
        //
        write(fd, buf, n); // :
    }
}
}

close(epoll_fd);
close(listen_fd);
return 0;
}

```

المثال على ملاحظات

- الجديدة. الاتصالات لتحديد EPOLLIN يتبع: الاستماع مرفق
- الواردة. البيانات لتحديد EPOLLIN أي ضا تتبع: العملاء مرفقات
- EPOLLOUT واستخدم الحافظة في البيانات بتخزين قم الإنتاج، في بالكامل. يكتمل write أن يفترض: تسوية الجزئية. للكتابة
- epoll. من وإزالة الملف نهاية أو الأخطاء عند المرفقات يغلق: الأخطاء معالجة

ملخص

تسجيل 2. `epoll_create1` باستخدام مثيل إنشاء 1. لينكس: في ملفات معالمة عدة مع لتعامل فعالة طريقة `epoll` يوفر في الفعاليات معالجة 4. `epoll_wait` باستخدام الفعاليات انتظار 3. `epoll_ctl` باستخدام والفعاليات الملفات معالمة الحاجة. حسب المعالمة إزالة أو تتبعتها يتم التي الفعاليات تعديل حلقة،

وضع استخدام في فكر الأداء، عاوية للاحتياجات بالنسبة المستوى. تتبع وضع باستخدام يوصى البسيطة، للتطبيقات أفضل على للحصول متزامن غير إدخال وإخراج مع `epoll` استخدام دائماً بعناية. المتاحه البيئات جميع معالجة مع الحافة تتبع الانتائج.