OS Lab 32

Status	ready
checkbox	☑
class	OS
due date	@Apr 21, 2021

Task

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/77f0a2b7-e8da-4b5f-ab48-2dfac51e657
1/_4.2_-_asynchio.pdf

Notes

Интерфейсы асинхронного ввода/вывода, определяемые стандартом POSIX, дают непротиворечивый способ асинхронного ввода/вывода, независимо от типов файлов. Эти интерфейсы заимствованы из предварительного стандарта реального времени, который включался как расширение в стандарт Single UNIX Specification. В версии Single UNIX Specification 4 эти интерфейсы перенесены в раздел базовых спецификаций, поэтому в настоящее время они обязательно должны поддерживаться всеми платформами.

Функции асинхронного ввода/вывода используют для описания асинхронных операций управляющие блоки AIO. Структура `aiocb` определяет управляющий блок AIO. Она содержит по меньшей мере следующие поля (реализации могут включать в структуру дополнительные поля):

```
struct aiocb {
  int             aio_fildes;     /* дескриптор файла */
  off_t           aio_offset;     /* смещение в файле */
  volatile void  *aio_buf;        /* буфер ввода/вывода */
  size_t          aio_nbytes;     /* количество байтов */
  int             aio_reqprio;    /* приоритет */
  struct sigevent aio_sigevent;   /* информация о сигнале */
  int             aio_lio_opcode; /* операция для списка запросов */
};
```

Поле `aio_fildes` — это дескриптор открытого файла, к которому применяется операция чтения или записи. Чтение или запись начинаются со смещения, определяемого полем `aio_offset`. При выполнении операции чтения данные копируются в буфер, начинающийся с адреса, определяемого полем `aio_buf`. При выполнении операции записи данные копируются из этого буфера. Поле `aio_nbytes` определяет количество байтов, которые нужно прочитать или записать.

Обратите внимание на необходимость явно указывать смещение при выполнении асинхронных операций ввода/вывода. Асинхронные функции ввода/вывода не оказывают влияния на позицию в файле, поддерживаемую операционной системой. В этом нет никакой проблемы, если в процессе никогда не смешиваются асинхронные функции ввода/вывода с их обычными аналогами. Отметьте также, что при записи в файл, открытый для записи в конец (с флагом `O_APPEND`), с использованием асинхронного интерфейса поле `aio_offset` в управляющем блоке AIO игнорируется системой.

Зачем нужен асинхронный в/в

- Создавать нить на каждую операцию ввода/вывода дорого
- Порядок исполнения нитей не гарантируется
- Некоторые устройства передают данные только после явного запроса (select/poll не подходит)
- Задержки при работе с такими устройствами чувствительны для приложений жесткого реального времени

Sigsetjmp/siglongjmp

```
#include <setjmp.h>

int sigsetjmp(sigjmp_buf env, int savemask);
void siglongjmp(sigjmp_buf env, int val);
```

- Siglongjmp сообщает среде исполнения, что мы вышли из обработчика сигнала (простой longjmp этого не делает).
- При выходе из обработчика сигнала восстанавливается маска сигналов процесса
- Используя параметр savemask, можно восстанавливать маску сигналов на момент sigsetjmp либо на момент прихода сигнала

Зачем использовать sigsetjmp/siglongjmp?

- Защита от ошибки потерянного пробуждения

```
aio_read(readrq);
        <<< прилетает сигнал
pause();
```

Другие поля никак не связаны с обычными функциями ввода/вывода. Поле `aio_reqprio` дает приложению возможность подсказать системе, в каком порядке должны выполняться асинхронные операции. Однако система дает лишь ограниченный контроль над порядком выполнения запросов, поэтому нет никаких гарантий, что значение этого поля будет учитываться системой в полной мере. Поле `aio_lio_opcode` используется только в списках запросов на асинхронный ввод/вывод, о которых рассказывается чуть ниже. Поле `aio_sigevent` определяет способ извещения приложения о завершении ввода/вывода. Этот способ описывается структурой `sigevent`.

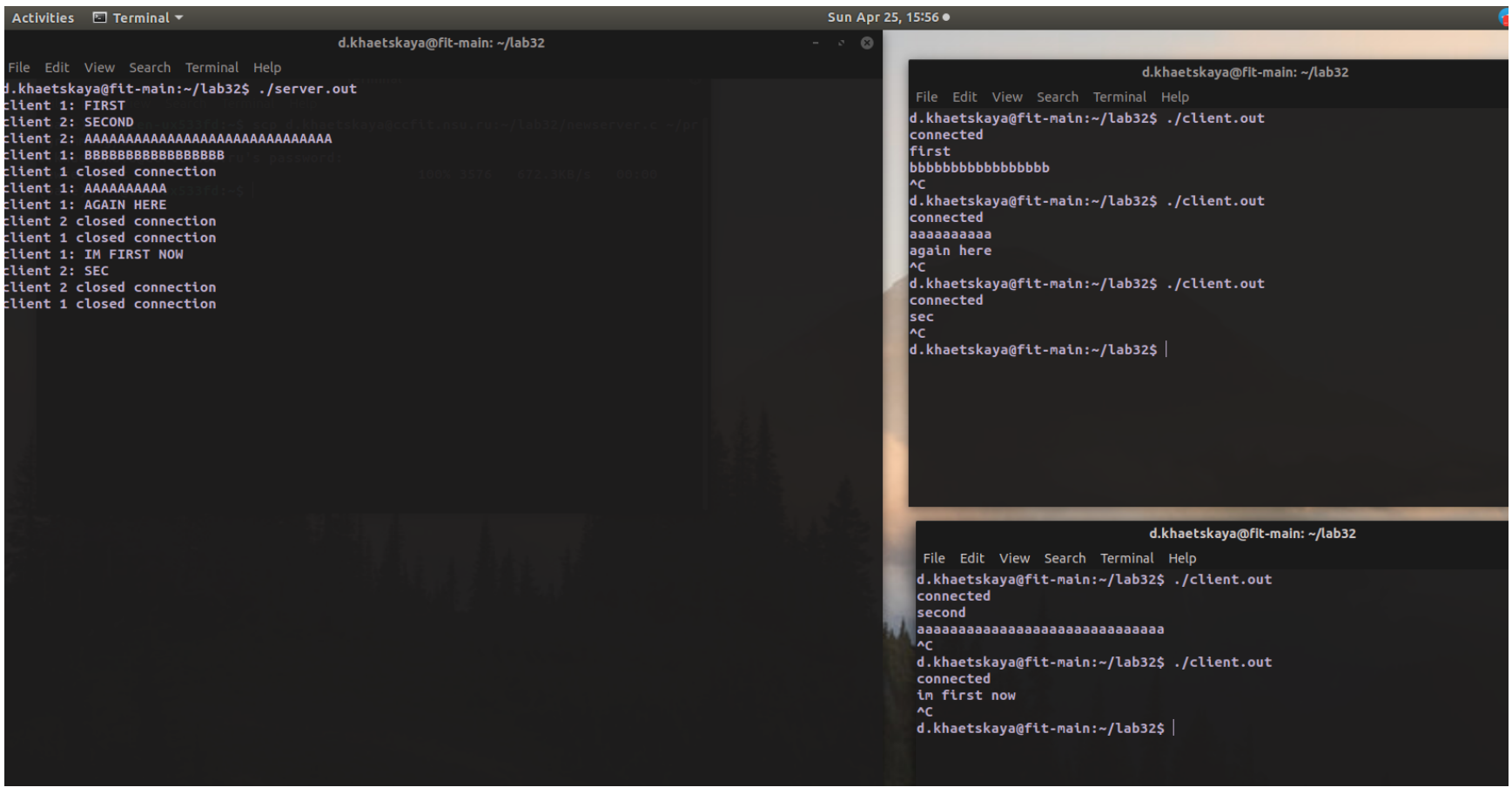
```
struct sigevent {
  int             sigev_notify;               /* тип извещения */
  int             sigev_signo;                /* номер сигнала */
  union sigval    sigev_value;                /* аргумент обработчика */
  void (*sigev_notify_function)(union sigval); /* функция-обработчик */
  pthread_attr_t *sigev_notify_attributes;    /* атрибуты обработки */
};
```

Поле `sigev_notify` определяет тип извещения. Оно может принимать одно из следующих трех значений:

`SIGEV_NONE` Процесс не извещается о выполнении асинхронной операции ввода/вывода.

`SIGEV_SIGNAL` По завершении асинхронной операции ввода/вывода процессу посылается сигнал, указанный в поле `sigev_signo`. Если приложение предусматривает обработку сигнала и установило флаг `SA_SIGINFO` при регистрации обработчика сигнала, сигнал будет поставлен в очередь (если реализация поддерживает такую возможность). Обработчик сигнала получит структуру `siginfo`, поле `si_value` которой будет хранить значение поля `sigev_value` (опять же, если был установлен флаг `SA_SIGINFO`).

https://illumos.org/man/3C/aio_read



Reading list

- [ ]