Submit a solution for mz17-1-KP. 2 - 1

Full score:	1
Time limit:	1 s
Real time limit:	5 s
Memory limit:	64M
Stack limit:	8M
Open date:	2022-12-19 10:30:00
Deadline:	2022-12-19 12:05:00

Problem mz17-1: KP. 2 - 1

В варианте файловой системы FAT12 под хранение информации о размещении блока отводится 12 битов. То есть для хранения информации о двух блоках требуется три байта: информация о первом блоке хранится в первом байте и 4 младших битах второго байта, информация о втором блоке хранится в старших битах второго байта и в третьем байте. По меньшим адресам хранятся младшие биты значения.

Номера блоков 0, 1, 0xFFE, 0xFFF не используются. Блок 0xFFF означает последний блок файла. Размер одного блока - 512 байтов.

В аргументах командной строки задаётся имя бинарного регулярного файла размером 6144 байта, содержащего таблицу FAT12 для файловой системы. Файловая система, описываемая таблицей, корректна.

На стандартном потоке ввода подаются пары чисел B0, OFF. Где B0 — начальный блок некоторого файла. OFF — смещение в файле (OFF >= 0).

На стандартный поток выведите номер блока на диске, по которому располагается байт с указанным смещением, и смещение байта в блоке (32-битные беззнаковые числа). Если смещение в файле слишком велико, выведите два числа 0.

Например, если цепочка блоков для некоторого файла имеет вид 1082, 2447, 3664 и смещение в файле равно 1000, то на стандартный поток вывода должно быть напечатано

```
2447 488
```

Пример файла с таблицей FAT можно скачать.

Submit a solution

Language: gcc-32 - GNU C (32 bit) 12.1.1

File Выберите файл Файл не выбран

Send! Send!

Submit a solution for mz17-2-KP. 2 - 2

Full score:	1
Time limit:	1 s
Real time limit:	5 s
Memory limit:	128M
Stack limit:	8M
Open date:	2022-12-19 10:30:00
Deadline:	2022-12-19 12:05:00

Problem mz17-2: KP. 2 - 2

Некоторый 32-битный процессор использует 26-битную виртуальную и физическую страничную адресацию памяти с размером страницы 1024 байта. В процессоре используется двухуровневая схема трансляции виртуального адреса в физический, в которой на каждый уровень таблицы страниц отводится по 8 битов. То есть номер в таблице страниц верхнего (0) уровня занимает биты 18-25 адреса, номер в таблице страниц 1 уровня занимает биты 10-17 адреса, и смещение в странице занимает биты 0-9 адреса.

Каждая запись в таблице страниц уровня 0 занимает 32 бита. Бит 0 — бит присутствия. Если он равен 1, то в битах 10-25 содержит номер физической страницы, в котором располагается соответствующая страница уровня 1. Значения остальных битов могут быть произвольными. Если бит присутствия равен 0, значения всех остальных 31 битов могут быть произвольными.

Каждая запись в таблице страниц уровня 1 занимает 32 бита. Бит 0 — бит присутствия. Если бит присутствия равен 0, то соответствующая виртуальная страница не отображена. Значения остальных 31 битов могут быть произвольными. Если бит присутствия равен 1, то в битах 10-25 содержится номер физической страницы, в которую отображается соответствующая виртуальная страница. Бит 1 равен 1, если страница доступна на запись, бит 2 равен 1, если страница доступна на выполнение. Отображенные страницы всегда доступны на чтение. Прочие биты могут иметь произвольное значение.

Напишите функцию:

```
int
lookup(
        void *memory,
        unsigned int pt,
        unsigned int va,
        int mode,
        unsigned int *pa);
```

Функция выполняет отображение виртуального адреса в физический. `pt` — физический адрес начала таблицы 0 уровня. Он всегда выровнен по адресу начала страницы. `pt` не может быть нулем. `va` — виртуальный адрес для отображения в физический. `mode` — режим доступа к памяти: 0 — чтение, 2 — запись, 4 — исполнение, 6 — запись и исполнение. Предполагается, что операция выполняется с одним байтом.

Функция должна вернуть 0, если виртуальная страница не отображена, либо отображенная страница не позволяет выполнить указанную операцию. В противном случае функция должна вернуть 1 и по адресу `pa` записать физический адрес, соответствующий виртуальному адресу `va`.

Параметр `memory` — это указатель на область памяти размером 64MiB, в которой находится содержимое физической памяти процессора в момент выполнения операции отображения.

Submit a solution

Language: gcc-32 - GNU C (32 bit) 12.1.1

File Выберите файл Файл не выбран

Send! Send!

View
all / necessary

up02-4
up02-5
up02-6
up02-7
mz09-1
mz09-2
mz09-3
mz09-4
mz09-5
mz09-6
up15-1
up15-3
up15-4
up15-5
up16-5
mz17-1
mz17-2
mz17-3
mz17-4
mz17-5

Submit a solution for mz17-3-KP. 2 - 3

Full score:	1
Time limit:	1 s
Real time limit:	5 s
Memory limit:	64M
Stack limit:	8M
Open date:	2022-12-19 10:30:00
Deadline:	2022-12-19 12:05:00

Problem mz17-3: KP. 2 - 3

На стандартном потоке ввода задается последовательность положительных целых чисел, которые задают "глубину" родства относительно главного процесса. То есть 1 обозначает сына главного процесса, 2 — его внука, 3 — правнука и т. д.

Первое число считывает главный процесс, он выводит на стандартный поток вывода свой PID и считанное число и создает цепочку процессов нужной глубины. Последний процесс в этой цепочке считывает со стандартного потока ввода очередное число, выводит на стандартный поток вывода свой PID и считанное число, после чего вся цепочка процессов до главного процесса завершается, а главный процесс идет на очередную итерацию и создает новую цепочку процессов глубины, значение которой было считано последним созданным процессом на этой итерации. Если был считан признак конца файла, то на стандартный поток вывода ничего не выводится, и все процессы, включая главный процесс, завершаются. Главный процесс должен завершаться с кодом 0.

Например, если дана последовательность 1, 2, 1, она обрабатывается следующим образом:

- Процесс отец считывает 1, печатает свой PID и число 1, после чего создает сына.
- Сын считывает 2, печатает свой PID и число 2, после чего завершает работу.
- Отец создает сына, сын внука, внук считывает 1, печатает свой PID и число 1, после чего внук и сын завершают работу.
- Отец создает сына, сын считывает конец файла, после чего завершает работу.
- Отец завершает работу.

Таким образом, вывод на стандартный поток будет примерно таким:

```
19596 1
19597 2
19599 1
```

Все числа во входном потоке больше 0 и меньше 115. Гарантируется что самая глубокая иерархия процессов может быть создана. Входной поток содержит как минимум одно число.

Submit a solution

Language: gcc-32 - GNU C (32 bit) 12.1.1

File Выберите файл Файл не выбран

Send! Send!

View
all / necessary

up02-4
up02-5
up02-6
up02-7
mz09-1
mz09-2
mz09-3
mz09-4
mz09-5
mz09-6
up15-1
up15-3
up15-4
up15-5
up16-5
mz17-1
mz17-2
mz17-3
mz17-4
mz17-5

Submit a solution for mz17-4-KP. 2 - 4

Full score:	1
Time limit:	1 s
Real time limit:	5 s
Memory limit:	64M
Stack limit:	8M
Open date:	2022-12-19 10:30:00
Deadline:	2022-12-19 12:05:00

Problem mz17-4: KP. 2 - 4

В аргументах командной строки программе передаются три параметра: `CMD1 FILE2 CMD3 CMD4` (в указанном порядке).

Реализуйте команду shell:

```
{CMD1 < FILE2 | CMD3 } && CMD4
```

Конвейер считается успешно завершившимся, если успешно завершился каждый процесс в составе конвейера.

Основной процесс должен дожидаться завершения выполнения созданных им процессов и только после этого завершаться с кодом возврата 0.

Для запуска процессов используйте `fork` и `execlp`. Использовать `system` или `/bin/sh` или аналогичные высокоуровневые средства запрещено.

Submit a solution

Language: gcc-32 - GNU C (32 bit) 12.1.1

File Выберите файл Файл не выбран

Send! Send!

View
all / necessary

up02-4
up02-5
up02-6
up02-7
mz09-1
mz09-2
mz09-3
mz09-4
mz09-5
mz09-6
up15-1
up15-3
up15-4
up15-5
up16-5
mz17-1
mz17-2
mz17-3
mz17-4
mz17-5

Submit a solution for mz17-5-KP. 2 - 5

Full score:	1
Time limit:	1 s
Real time limit:	1 s
Memory limit:	64M
Stack limit:	8M
Open date:	2022-12-19 10:30:00
Deadline:	2022-12-19 12:05:00

Problem mz17-5: KP. 2 - 5

Морской бой под Воронежем.

В аргументах командной строки задается число N процессов (N > 0), и число жизней у каждого процесса L (L > 0).

Главный процесс должен создать N процессов, пронумерованных от 0 до N - 1. Далее главный процесс считывает со стандартного потока ввода последовательность 32-битных беззнаковых чисел. Номер процесса определяется как остаток от деления считанного числа на N. Главный процесс "стреляет" в этот процесс. Соответствующий процесс выводит на стандартный поток вывода свой номер и количество оставшихся жизней. Если процесс “убит”, то есть у него не осталось жизней, процесс завершается. Выстрел в убитый процесс не выполняет никаких действий.

В конце ввода со стандартного потока ввода отец выводит число оставшихся в живых процессов, после чего все оставшиеся процессы должны завершиться, после чего завершается отец.

Для синхронизации процессов можете использовать любой корректный способ.

Например, если программа запущена с аргументами:

```
./solution 2 3
```

и на стандартный поток ввода подается

```
0 1 1 1
```

то на стандартный поток вывода должно быть напечатано

```
0 2
1 2
1 1
1 0
1
```

Submit a solution

Language: gcc-32 - GNU C (32 bit) 12.1.1
File Выберите файл Файл не выбран
Send! Send!

View
all / necessary
up02-4
up02-5
up02-6
up02-7
mz09-1
mz09-2
mz09-3
mz09-4
mz09-5
mz09-6
up15-1
up15-3
up15-4
up15-5
up16-5
mz17-1
mz17-2
mz17-3
mz17-4
mz17-5