Домашнее задание по ОС №2

Me

November 29, 2016

Contents

1 Основное задание

В этом домашнем задании вам нужно реализовать файловую систему в памяти (что довольно просто, но не забывайте про синхронизацию) и заполнить ее начальными данными (initramfs, о которой далее сказано подробней).

1. Реализовать файловую систему в памяти, которая поддерживает следующий минимальный набор операций:

   - open/close - создание/открытие и закрытие файла;
   - read/write - чтение и запись файла по указанному смещению;
   - mkdir - создание каталога;
   - readdir - перечисление записей каталога;

   Конкретный набор функций реализующих эти операции остается на ваше усмотрение. Хорошим вариантом будет мимикрировать Unix API для работы с файловой системой.

2. Распарсить образ initramfs в формате cpio в памяти и заполнить его содержимым файловую систему;

Для простоты считайте, что все имена файлов и каталогов даются в абсолютном формате, т. е. такого понятия как текущий каталог у нас нет.

2 Initramfs

Initramfs (Initial RAM FS) - начальная файловая система в памяти. Обычно она используется только при старте ОС, чтобы ОС могла найти на ней исполняемый файл первого приложения (init) и утилиты нужные для конфигурации системы.

В нашем случае начальная файловая система так же будет и единственной, т. е. другой файловой системы кроме этой у нас не будет.

Стоит отметить, что не все системы используют initramfs, некоторые системы могут сразу читать данные с диска. Но мы не будем так делать, потому что реализация файловой системы в памяти гораздо проще.

2.1 Формат CPIO

Итак откуда начальная файловая система берет свое содержимое? Обычно содержимое запаковано в некоторый архив[1] в специальном формате, который загружается в память системы вместе с ядром ОС. О том куда именно он загружается мы поговорим дальше, сейчас мы сосредоточимся на формате архива.

Вам в домашнем задании предлагается использовать бывший некогда популярным формат cpio[2]. Формат cpio очень прост:

- каждая сущность (файл/каталог) в архиве начинается со специального заголовка фиксированного формата;
- каждый заголовок выровнен на границу 4 байт, т. е. если размер описания предыдущей сущности не был кратен 4 (заголовок + имя + данные файла), то вы должны сами выровнять текущую позицию от начала архива;
- сразу за заголовком идет имя файла/каталога переменной длинны (размер имени есть в заголовке);
- далее, выровненными на границу 4 байт, идут данные файла, если заголовок описывал файл (размер данных опять же в заголовке); обратите внимание, что даже если длина имени не кратна 4 байтам, позиция данных файла все равно должна быть выровнена на границу 4 байт - ваша обязанность следить за этим при чтении;

Конец архива определяется либо когда вы не смогли прочитать сущность из архива целиком, либо когда в наткнулись на сущность с именем "TRAILER!!!"[3].

Чтобы сделать вашу жизнь еще "проще", все данные в заголовке записаны в текстовом представлении в 16-ти ричном формате[4]. Формат заголовка вы можете найти в файле initramfs.h.

Гарантируется, что каталоги встретятся в cpio архиве раньше, чем файлы в этих каталогах, если каталоги не пустые. Т. е. просматривая архив по порядку вы всегда сначала создадите каталог, а уже потом будете добавлять в него файлы.

[1] Не для сжатия, а для того чтобы был один файл.

[2] cpio - аналог tar, но имеет несколько более простое внутреннее представление.

[3] Ппц...

[4] Без всяких префиксов, наиболее значимые цифры по меньшим индексам массива - детали легко определяются экспериментально

Для того чтобы распарсить этот формат остается одна маленькая деталь - нужно уметь отличать каталоги от файлов - для этого в initramfs.h определены макросы:

- S_ISDIR - проверяет, что число записанное в поле mode заголовка соответствует каталогу;
- S_ISREG - проверяет, что число записанное в поле mode заголовка соответствует обычному файлу[1].

2.2 Создание образа initramfs

Для создания образа initramfs вам предоставлен bash скрипт make_initramfs.sh. Как первый аргумент он принимает имя каталога, который нужно запаковать в initramfs[2]. Вторым параметром скрипт получает имя выходного файла, в который он запишет этот образ. Для создание образа используется утилита cpio.

2.3 Передача образа initramfs

Теперь нам нужно каким-то образом передать этот образ в QEMU, чтобы он был загружен вместе с ядром ОС. А кроме того внутри ядра ОС каким-то образом узнать куда этот образ будет загружен.

Первая часть решается легко. У QEMU есть опция -initrd, после которой нужно указать имя файла.

Со второй частью все несколько сложнее. Так как наше ядро пользуется спецификацией multiboot разумно искать в ней указания на то, где нужно искать образ initramfs. Но, к сожалению, в спецификации multiboot нет ничего подобного.

Но в multiboot есть такая вещь как загружаемые модули. Собтственно QEMU передает образ initramfs под видом одного из этих модулей. За модули в заголовке multiboot отвечают поля mods_count и mods_addr. Но перед тем как обращаться к ним, нужно убедиться, что они содержат валидную информацию. Для этого нужно проверить установлен ли 3-ий бит[3] в поле flags заголовка multiboot.

Если информация в этих полях валидна, то mods_addr содержит адрес массива дескрипторов описывающих модули. Конкретный формат дескриптора уточняйте в документации multiboot.

[1] "Необычные" файлы нас не интересуют.

[2] Учтите, что это имя будет частью имени сущности в архиве.

[3] Считая с 0.

Как найти среди этих модулей тот, что нужен нам? Это довольно легко, если мы знаем где располагается модуль в памяти. Как известно initramfs образ будет начинаться с заголовка cpio. Первое поле в каждом заголовке cpio это магическое значение. Т. е. все что нам нужно, так это проверить, что размер модуля достаточно большой и что он начинается с некоторой магической последовательности байт, а именно "070701".

Остается один небольшой момент, как и с ядром ОС вы должны указать вашим алокаторам памяти, чтобы они не могли алоцировать образ initramfs до тех пор пока вы не прочитали его.