17. Отображаемая память

Отображаемая память позволяет различным процессам общаться через общедоступный файл. Отображаемая память может использоваться для взаимодействия процессов или как простой способ для обращения к содержимому файла. Отображаемая память формирует ассоциацию между файлом и памятью процесса. Linux разбивает файл на куски размером страницы и затем копирует их в страницы виртуальной памяти так, чтобы они могли быть представлены в адресном пространстве процесса. Таким образом, процесс может читать содержание файла обычным доступом к памяти. Он может также изменить содержимое файла, записывая в память. Что позволяет быстро взаимодействовать с файлом.

17.1. Отображение обычного файла

Чтобы отобразить обычный файл в память процесса используйте вызов mmap (Memory MAPped , произносимое "em-map"). Первый параметр - адрес, начиная с которого Вы хотели бы, чтобы Linux отобразил файл; значение NULL заставляет Linux выбирать первый доступный адрес. Второй параметр - длина отображения в байтах. Третий параметр определяет защиту для отображаемого адресного интервала. Защита состоит из поразрядного "или" из PROT_READ, PROT_WRITE , и PROT_EXEC, что соответствует чтению, записи, и разрешению на выполнение. Четвертый параметр - значение флажка, определяющее дополнительные опции. Пятый параметр - дескриптор открытого файла, который будет отображен. Последний параметр - смещение от начала файла, с которого начнется отображение. Вы можете отобразить весь или часть файла в память, выбирая начальное смещение и длину.

Значение флажка - поразрядное "или" этих ограничений:

- MAP_FIXED - отображать только начиная с указаного адреса, если не удастся - вернуть ошибку, а не искать подходящий. Этот адрес должен быть выровнен на границу страницы.
- MAP_PRIVATE - производить запись изменений не в отображенный файл, а в его копию. Никакой другой процесс не увидит, изменений в файле. Этот режим не может использоваться с MAP_SHARED.
- MAP_SHARED - изменения в памяти немедленно отражаются в основном файле вместо буферизации. Используйте этот режим для межпроцессового взаиодействия. Не должно быть использованно вместе с MAP_PRIVATE.

Успешный вызов возвращает указатель на начало памяти. При ошибке, возвращает MAP_FAILED . Когда вы закончили работу с управлением памятью, освободите ее при помощи munmap. Передайте адрес начала и длину области отображаемой памяти. Linux автоматически освобождает отображаемую память при завершении процесса.

17.2. Пример программы

Давайте посмотрим на две программы демонстрирующие использование области отображаемой памяти, с целью чтения и записи в файлам. Первая программа пишет его в файл строку «Hello from mmap!». Вторая программа, листинг 2, читает ее, и выводит на экран. Обе программы считывают имена отображаемых файлов из командной строки.

Listing 1 (mmap-write.c) Записывает строку в отображаемую память.

```
#include <stdlib.h>
#include <stdio.h>
#include <fcntl.h>
#include <sys/mman.h>
```

```
#include <sys/stat.h>
#include <time.h>
#include <unistd.h>
#define FILE_LENGTH 0x100
int main (int argc, char* const argv[])
{
 int fd;
 void* file_memory;
 /* Открываем(создаем) файл, достаточно  большой, чтобы хранить нашу строку */
 fd = open (argv[1], O_RDWR | O_CREAT, S_IRUSR | S_IWUSR);
 lseek (fd, FILE_LENGTH+1, SEEK_SET);
 write (fd, "", 1);
 lseek (fd, 0, SEEK_SET);
 /* Создаем отображение в памяти. */
 file_memory = mmap (0, FILE_LENGTH, PROT_WRITE, MAP_SHARED, fd, 0);
 /* Пишем строку в отображенную память. */
 sprintf((char*) file_memory, "%s\n", "Hello from mmap!");
 /* Освобождаем память. */
 munmap (file_memory, FILE_LENGTH);
 close (fd);
 return 0;
}
```

Программа mmap-write открывает файл, создавая его, если он прежде не существовал. Третий параметр указывает режим доступа для чтения и записи. Мы используем lseek , чтобы гарантировать, что файл является достаточно большим, чтобы сохранить строку . После чего возвращаемся к началу файла.

Программа отображает файл и пишет строку в отображаемую память, таким образом и в файл, затем освобождает отображаемую память. Вызов munmap ненужен, потому что Linux автоматически освободил бы отображаемую память и файл, когда программа завершится.

Listing 2 (mmap-read.c) Читает строку из файла отображенного в памяти и выводит на консоль.

```
#include <stdlib.h>
#include <stdio.h>
#include <fcntl.h>
#include <sys/mman.h>
#include <sys/stat.h>
#include <unistd.h>
#define FILE_LENGTH 0x100
int main (int argc, char* const argv[])
{
  int fd;
  void* file_memory;
  /* Открыть файл. */
  fd = open (argv[1], O_RDWR, S_IRUSR | S_IWUSR);
  /* Отобразить файл в память.  */
  file_memory = mmap (0, FILE_LENGTH, PROT_READ | PROT_WRITE,MAP_SHARED, fd, 0);
  printf ("%s\n", (char*) file_memory);
  /* Освобождение памяти. */
  munmap (file_memory, FILE_LENGTH);
  close (fd);
  return 0;
}
```

Программа mmap-read читает строку из файла и затем выводит его на консоль. Вот пример запуска примеров программ. Здесь отображаем файл test.

```
$ ./mmap-write test
$ cat test
Hello from mmap!
```

```
$ ./mmap-read test
Hello from mmap!
```

17.3. Совместный доступ к файлу

Различные процессы могут взаимодействовать используя области отображаемой памятью, связанные с одним и тем же файлом. Укажите флажок MAP_SHARED для того, чтобы любые операции записи в область памяти немедленно передаются файлу и видимым другим процессам. Если Вы не определяете этот флажок, Linux может буферизовать операции записи перед передачей их к файлу.

С другой стороны вы можете заставить Linux синхронизировать буфер с файлом на диске, вызвав msync . Его первые два параметра определяют область отображенной памяти, как и для munmap . Третий параметр может содержать следующие значения флажка:

- MS_ASYNC - обновление намечается, но не обязательно выполненныется перед возвращением управления.
- MS_SYNC - обновление непосредственно; вызов не возвращет управление пока не завершит синхронизацию. MS_SYNC и MS_ASYNC не должны прсутствовать вместе.
- MS_INVALIDATE - все другие отображения файла изменены так, что бы они могли видеть модифицированные значения.

Например, чтобы сбросить на диск буфер общедоступного файл, отображенный в адресе mem_addr длины mem_length байт:

```
msync (mem_addr, mem_length, MS_SYNC | MS_INVALIDATE);
```

Как и с совместно используемой памятью, пользователи областей отображенной памяти должны следовать протоколу, чтобы избежать условий гонки. Например, семафор может использоваться, чтобы препятствовать доступу более одного процесса к отображенной памяти.

17.4. Частные отображения

Указывая флаг MAP_PRIVATE при вызове mmap создается область копирования при записи. Любая операция записи отражается только в памяти этого процесса; другие процессы, которые отображают тот же самый файл, не будут видеть изменения. Вместо того, чтобы писать непосредственно странице, разделенной всеми процессами, процесс пишет частной копии этой страницы. Все последующие операции читения и записи процессом используют эту же страницу.

17.5. Другие применения mmap

Вызов mmap может использоваться и в целях не связанных с взаимодействием процессов. Одно из применений - замена операций для чтения и записи. Например, вместо того, чтобы явно читать содержимое файла в память, программа могла бы отобразить файл в память и просмотреть его используя чтение памяти. Для некоторых программ, это более удобно и может работать быстрее, чем явные операции ввода - вывода.

Одна мощная методика, используемая некоторыми программами - формирование структуры данных в файле отображенной памяти. При следующем запуске, программа отображает тот файл назад в память, и структуры данных восстановлены в их предыдущеее состояние. Обратите внимание, что указатели в этих структурах данных будут недопустимы, если они все не указывают в пределах той отображенной области памяти или если файл отображен по тем же адресам, которые занимал первоначально.