Предварительный проект микроядра с рабочим названием «Изделие 0»

Загрузка системы

1.1 Начальный загрузчик (boot loader) загружает и запускает вторичный загрузчик.

1.2 Вторичный загрузчик реализует:
1.2.1 Поиск устройств PCI и загрузку их драйверов (группа процессов drivers);
1.2.2 Поиск существующих структур и загрузку серверов структур логических абстракций (группа процессов structures);
1.2.3 Загрузку процессов системные функции (группа процессов system);
1.2.4 Загрузку процесса начальной инициализации системы (особая группа, в которую входит только один процесс, запуск которого имеет право только ядро и только в момент начальной загрузки)
1.2.5 Загрузку и запуск ядра (загрузка по выбранному адресу, включение защищенного режима, передача управления на код ядра).

Уточнения:
1. Вторичный загрузчик может работать как в защищенном режиме процессора, так и реальном. Скорее всего, придется использовать оба режима, поскольку драйвер устройства, с которого будет производиться загрузка, легче реализовать в реальному режиме, как и поиск PCI устройств, воспользовавшись системными вызовами BIOS, а вот пересылку данных за пределами первого мегабайта проще организовать, а защищенном режиме. Хотя для простоты можно установить нереальный режим работы процессора, но это не лучший выход, поскольку производители процессоров могут и не поддержать эту особенность.
2. Вторичный загрузчик размещается в области не критичной для функционирования запускаемого ядра и затирается при старте ядра. Память, где он располагался, может использоваться ядром без ограничений.

1.3 Ядро при своем запуске активирует процесс инициализации, который выполняется в нулевом кольце защиты и на время выполнения берет все управление на себя. В его задачи входит:
1.3.1 Очистка критичных рабочих областей;
1.3.2 Установление связей между ядром и загруженными драйверами, структурами и ядром, в том числе распределение прерываний между устройствами и драйверами;
1.3.3 Запуск процесса отвечающего за загрузку и запуск пользовательских процессов: сервера API и рабочая среда (текстовая оболочка по типу bash или графическая любого типа).

1) Скелетная схема функционирования ядра.

Большинство функций ядра вынесено за его пределы и реализуются при помощи серверов. Логически есть несколько уровней между процессами, физически было бы лучше использовать все 4 уровня защиты, изначально предполагающиеся в процессоре, но поскольку для отдельных видов функций оставлено только два уровня защиты, то придется отказаться от этого в пользу двухуровневой системы.
Уровень 0 - само ядро с функциями запуска и убиения процессов, обмена сообщениями между процессами (IPC);
Уровень 1 – системные функции: менеджер памяти, планировщик процессов и др. Группа процессов system. Вероятно, Уровень 0 и Уровень 1 будут объединены в одном уровне ядра, по крайней мере, для первой реализации.

Уровень 2 – Драйвера физических устройств. Группа процессов drivers
Уровень 3 – Структуры логических абстракций: файловые системы, включая VFS, функции графических примитивов и др. Группа процессов structures
Уровень 4 – Приложения запускаемы пользователем и приложения запускаемые системой для обеспечения работы с нестандартным оборудованием: съемные диски, периферийные устройства и др. Группа процессов users.

Уточнения:
Уровень 0 выполняется в кольце защиты Ring 0, все остальное функционирует в пределах Ring 1-3, при реализации четырехуровневой защиты, либо Ring 3 при реализации двухуровневой защиты.

2) **Обмен сообщениями между процессами – IPC**

Поскольку IPC один из важнейших элементов влияющих на быстродействие системы с микроядром (если не самый важный), то соответственно от его реализации зависти очень много.

Сокращения:
ПС - Процесс-Сервис
ПК - Процесс-Клиент
СПС - Семафор обработки Процесса-Сервиса
СПК - Семафор обработки Процесса-Клиента
ТСО - Таблица Служебных Областей
БДСО - Блок Данных Служебной Области

Ядро обеспечивает:
1. Получение содержимого ТСО.
2. Получение и отправка данных в область слотов указанного ПС.
3. Выделение, освобождение, монтирование и демонтирование служебной области для ПК или ПС, в соответствии с правами доступа, с размещением данных в ТСО.
4. Установка прав доступа к собственной служебной области ПК и ПС.
5. Выделение, освобождение и перераспределение обычной памяти для процессов.
6. Установка и удаление обработчиков прерываний.
7. Выделение и освобождение доступа к портам.

Механизм взаимодействия IPC следующий:
1. ПС запрашивает область памяти у менеджера памяти, под область слотов ПС. Область памяти заносится в ТСО, права доступа к области слотов (кроме ПС) имеет только ядро.
2. ПК запрашивает область памяти у менеджера памяти, под область данных ПК. Область памяти заносится в ТСО, права доступа к области данных (кроме ПК) имеет ядро и вызываемый ПС (в последствии).
3. ПК запрашивает ТСО.
4. ПК устанавливает для ПС права доступа к своей служебной области данных обозначенной в ТСО.
5. ПК запрашивает данные области слотов ПС.
6. ПК передает, через сервис ядра, в область слотов ПС свои рабочие данные, данные области данных ПК и выставляет СПС в значение 1 (запрос обработки), а также выставляет СПК в значение 1.
7. ПС последовательно обрабатывает все слоты. При приеме текущего запроса монтирует в свое пространство область данных ПК. Выставляет СПК в значение 2 -

запрос обрабатывается. После завершения обработки в область данных ПК размещаются данные и в СПК значение 3 (запрос обработан правильно) или значение 4 (запрос обработан с ошибкой).

В случае значений 3 и 4 – выставляется значение 0 в СПС. Область данных ПК демонтируется из области ПС.

8. ПК по мере переключения приоритета CPU на него проверяет СПК.

8.1 Если значение 0, то слот запроса пуст. Бывает только при старте процесса или принудительном сбросе ПК.

8.2 Если значение 1, то запрос в очереди, передать управление следующему процессу.

8.3 Если значение 2, то запрос обрабатывается, передать управление следующему процессу.

8.4 Если значение 3, то запрос обработан правильно.

8.5 Если значение 4, то запрос обработан с ошибкой.

В случае значений 3 и 4, ПК забирает права доступа ПС к своей служебной области данных.

ТСО имеет следующую структуру:

Заголовок - 32 байта
БДСО 0
БДСО 1
…
БДСО n

Заголовок:

- +0: dword: версия ТСО (текущая версия = 1)
- +4: dword: количество размещённых БСО; не больше, чем запрошено; может быть меньше, если в структуре закончились БСО.
- +8: dword: общее число БДСО в ТСО
- +12 = +0xC: 20*byte: зарезервировано (нули)

БСО:
Размер блока 32 байта

- +0: dword: номер идентификатор
- +4: dword: тип структуры: 1 – область слотов ПС, 2 – область данных ПК

Область слотов ПС имеет следующую структуру:

Заголовок - 32 байта
Слот 0
Слот 1
…
Слот n

Заголовок:

- +0: dword: указатель на текущий свободный слот (при значение -1, свободных слотов нет, ПК ожидает освобождения периодически проверяя)
- +4: dword: размер слота
- +8 и далее зарезервировано

Слот:

- +0: dword: СПС - Семафор обработки Процесса-Сервиса
- +4: dword: указатель на номер идентификатор в БДСО для возврата результата ПК
- +8 и далее определяется самим ПС

Область данных ПК имеет следующую структуру:
Заголовок – 32 байта
Данные задания
Данные результата:

Заголовок:

- +0: dword: СПК - Семафор обработки Процесса-Клиента
- +4: dword:
  Код ошибки
  0 – без ошибок
  1 – входные данные не верны
  и т.п. будет определено в дальнейшей проработке в зависимости от ПС
- +8 dword: Размер области под данные задания
- +12 dword: Размер области под данные результата
- +16 и далее зарезервировано

Данные задания и результата:
Содержимое определяется ПС к которому обращались.

3) Работа планировщика заданий (активности процессов).
Раздел в разработке (обдумывание).