Дорожная карта osFree

Разработка osFree должна делиться на несколько крупных этапов – от базового функционала с поддержкой командной строки к полнофункциональному работоспособному Workplace Shell]. Наше текущее состояние проекта помечено жирным шрифтом. === Статус готовности различных компонентов === * [[en:filesystems_support|Поддержка файловых систем

Этапы пути к версии 1.0

0.1

К этой версии функциональность реализации CPI должна быть, в основном, завершена. Поддержка 16-битных API на данном этапе не требуется. CMD.EXE и прочие утилиты должны быть пересобраны для получения полностью 32-битных приложений (т.к. только совместимость уровня исходного кода требуестя на данном этапе).

Версия	Требования	Статус
0.0.1	должна работать загрузка L4 при помощи GRUB и запускаться “Hello, world” (как L4 root task). “Hello, world” имитирует “ядро” osFree.	Готово.
0.0.2	Поддержка EXT2FS должна быть (в основном) закончена и должна работать последовательность загрузки. На данном этапе мы должны избавиться от GRUB и заменить его на наш собственный загрузчик. 16-битный MicroFSD (OS/2-совместимый), IFS 32-итная (не OS/2-совместимая). MicroFSD должен загружать и стартовать FreeLDR. FreeLDR стартует ядро L4 и root task (имиирующую ядро osFree). LILO на данном этапе не требуется для EXT2FS.	Готово.
	Замечание: Поддержка MiniFSD отстутствует, т.к. оно не требуется для случая загрузки L4.	
0.0.3	“Ядро” osFree должно прочитать и обработать CONFIG.SYS, используя file provider (на данном этапе пока не нужна реальная загрузка драйверов) и показать дерево настроек.	Готово.
0.0.4	LX-загрузчик и компоновщик исполняемых модулей (с использованием file provider-а). “Ядро” osFree должно загрузить и запустить приложение, указанное в PROTSHELL.	Готово.
0.0.5	Реализация файловых API. LX-loader должен стартовать задачу MINICMD.EXE, используя L4VFS. Должен быть реализован Минимальный набор API. Все операции с указателями должны быть заменены хендлами (нужен менеджер хендлов). OS/2 сервер должен стать более структурированным. Поддержка обычной для OS/2 раскладки адресного пространства. **Далее...**	**Разрабатывается**
0.0.6	Разрабатывается VIO сервер. VIO API (32-битная версия, на основе VIO сервера). Работа через l4 console вместо вывода на log server. Консоль на основе nitpicker (написать сервер типа proxygon).	Не готово
0.0.7	Разработка KBD API (32-битная версия)	Не готово
0.0.8	Параллельное исполнение нескольких приложений. Многонитевые приложения.	Не готово

Версия	Требования	Статус
0.0.9	OpenWatcom и утилиты для сборки должны быть пересобраны в виде 32-битных приложений (не должно быть зависимостей от 16-битных API). (LX формат)	Не готово
0.0.10	CMD.EXE должен быть собран под osFree и все нужные CPI должны быть реализованы. (LX формат, только 32-bit API)	Не готово
0.0.11	Сборка osFree под самой osFree. (LX формат, без 16-бит API)	Не готово
0.0.12	Open Object REXX должен быть перенесен под osFree. (LX формат, без 16-бит API)	Не готово

0.2

Должна быть реализована поддержка16-бит API.

Версия	Требования	Статус
0.1.1	Должна быть реализована поддержка16↔32 thinking	Не готово
0.1.2	16-битные обертки 32-битных API	Не готово
0.1.3	Поддержка ELF формата	Не готово
0.1.4	Поддержка формата NE	Не готово

0.3

Поддержка SOM и SOM toolkit. (Использовать код NOM???)

Версия	Требования	Статус
0.2.1	SOM Compiler Watcom Linker Emitter	Готово
0.2.2	SOM Compiler Preprocessor	Готово
0.2.3	Должен быть реализован SOM Compiler C Emitter	Не готово
0.2.4	Должен быть реализован SOM Compiler C++ Emitter	Не готово
0.2.5	Должны быть разработаны Emitter Framework classes	Не готово
0.2.6	Должен быть готов SOM Compiler. (Поддержка последних версий IDL)	Не готово
0.2.7	SOM обертка для CPI (CPI+)	Не готово

0.4

Поддержка GPI (Пока без SOM)

0.5

Разарботка PM (PM на основе SOM???)

0.6

Разработка WPS

0.7

Сетевой стек