Первые шаги

Перед тем как начать что-то кодировать, почитайте исходники и определите область, которая вас заинтересует для разработки. У нас есть очень широкий круг задач и, очень возможно, одна из них вас заинтересует. Только помните, что наша текущая главная задача – реализовать более или менее полный набор OS/2 API поверх микроядра L4.

Инструменты разработки

Язык программирования	Компилятор
C	Open Watcom
C++	Open Watcom
FORTRAN	Open Watcom
Pascal	FreePascal
REXX	ReginaREXX

Другие языки тоже могут быть использованы для разработки частей ОС. Но они тоже должны быть Open Source и использовать OS/2 API. Использование платных и shareware (то есть, которые могут быть скачаны с многих мест, но стоят денег и лицензия которых имеет много ограничений) не рекомендуется. Желательно использовать инструменты из поставки OpenWatcom для сборки ваших исходников.

Когда существуют аналогичные инструменты (например NMAKE и WMAKE), всегда используйте инструменты из поставки OpenWatcom.

Получение и сборка исходных кодов

Сначала, вы должны загрузить нижеперечисленные инструменты для вашей платформы с соответствующих сайтов:

- http://www.openwatcom.org/ для OpenWatcom
- http://www.freepascal.org/ для FreePascal
- http://regina-rexx.sf.net/ для ReginaREXX

Вы можете загрузить нерегулярно обновляемые снапшоты исходников с этого сайта, или самые последние версии с SVN. Исходники osFree находятся на Sourceforge SVN. Sourceforge также позволяет скачать любую ревизию в виде .tar.gz архива.

Перед сборкой проверьте файлы setvars-<somename>.cmd и <somename>.conf, И поменяйте настройки (в основном, пути к инструментам разработки). После этого откройте сеанс командной строки и запустите setvars-<somename>.cmd и введите

```
wmake
```

начнется процесс сборки. Для очистки дерева исходников от созданных при сборке файлов наберите

```
wmake clean
```

Для более подробной информации о системе сборки см. документ Система сборки.

Дерево каталогов

Посмотрите на код в SVN, для понимания принципа размещения файлов. Обратите внимание, что дерево файлов в SVN osFree состоит из исходных кодов операционной системы и инструментов тулкита. Пожалуйста, НЕ помещайте сюда приложения и инструменты, не соответствующие назначению тулкита. Тулкит это вспомогательные утилиты, собираемые под ту систему, под которой ведется сборка и необходимые для сборки файлов ОС.

Global/Shared/Private Headers

Each level of the SVN tree contains two standard directories:

```
FIX THIS! Now it's slightly different!
```

shared	Contains code shared among all source at this level and deeper levels
include	Contains header files for the above

Each levels/part of the OS should have a specific prefix that allows a developer to easily find what part of the OS a header/library file belongs to. For example code shared by the whole tree should be included with:

```
#include <all_shared.h>
```

and code shared by all commandline tools should include:

```
#include <cmd_shared.h>
```

Try to create as few shared code headers as possible. Each “shared” directory should contain one (1) library (.lib) file (xxx_shared.lib) with all shared code and each “include” directory should contain one main header file including all other (xxx_shared.h).

Example of use of common files:

```
// Use the normal OS/2 INCL_ since our toolkit is the OS/2 toolkit
#define INCL_DOSERROR

// Do NOT include os2.h, use osfree.h instead
#include <osfree.h>

// Include any needed normal C library
#include <malloc.h>
#include <string.h>

// Include all shared code and shared code for command line tools
#include <all_shared.h>
#include <cmd_shared.h>
```

Documentation

- Private code (not shared with other code) should be documented only in the source.
- Shared code (shared with code at the same level or at all levels) should be documented in source and in a “building and developing” document.
- The API of the OS and the tools documentation should NOT be documented in the source tree but in the toolkit and release tree.
- Source code should be documented in the source file (not the header files).
- Each function should be prefixed with a description of what it does, what parameters it uses (in and out) and any external references it uses.
- Place comments in the source that helps the reader to understand the logic and don't overdo it.

When Developing

- Use static linking, do not use dynamic libraries (LIBC style) or dynamic runtime.
- Use the makefiles provided with the source tree, don't “do your own”.
- Currently osFree development is done on OS/2 (minimum Warp 4) but in the future development will be hosted on osFree.
- We use SVN to share code among developers.
- We use Doxygen and Wiki to document our work.

Submitting a Patch (FIX THIS!!!)

- Make sure your changes follow the coding guidelines above.
- Make sure you are using the current versions of the sources so that the resulting diffs are comparing your changes with the head of the source tree.
- Create your patch either by using cvs diff -u (if you are using CVS) or diff -u original-file changed-file (if you are using a source archive - you can also create differences for the whole directory contents using diff -r) In the latter case include the old code first, the new code last – in the patch anything you added will be prefixed with a +.
- Remove all/any lines that reference files without changes.
- Send the patch file as an attachment in your email. Do not paste the patch directly into the email body.
- Maintainers will often reply in response to your patch, pointing out things to fix up, etc. before a patch can be checked in. Please always follow the maintainer suggestions closely and respond by sending a new corrected patch. Please do not expect the maintainers to rework your changes, you want to be able to claim all the credit for your great patches!