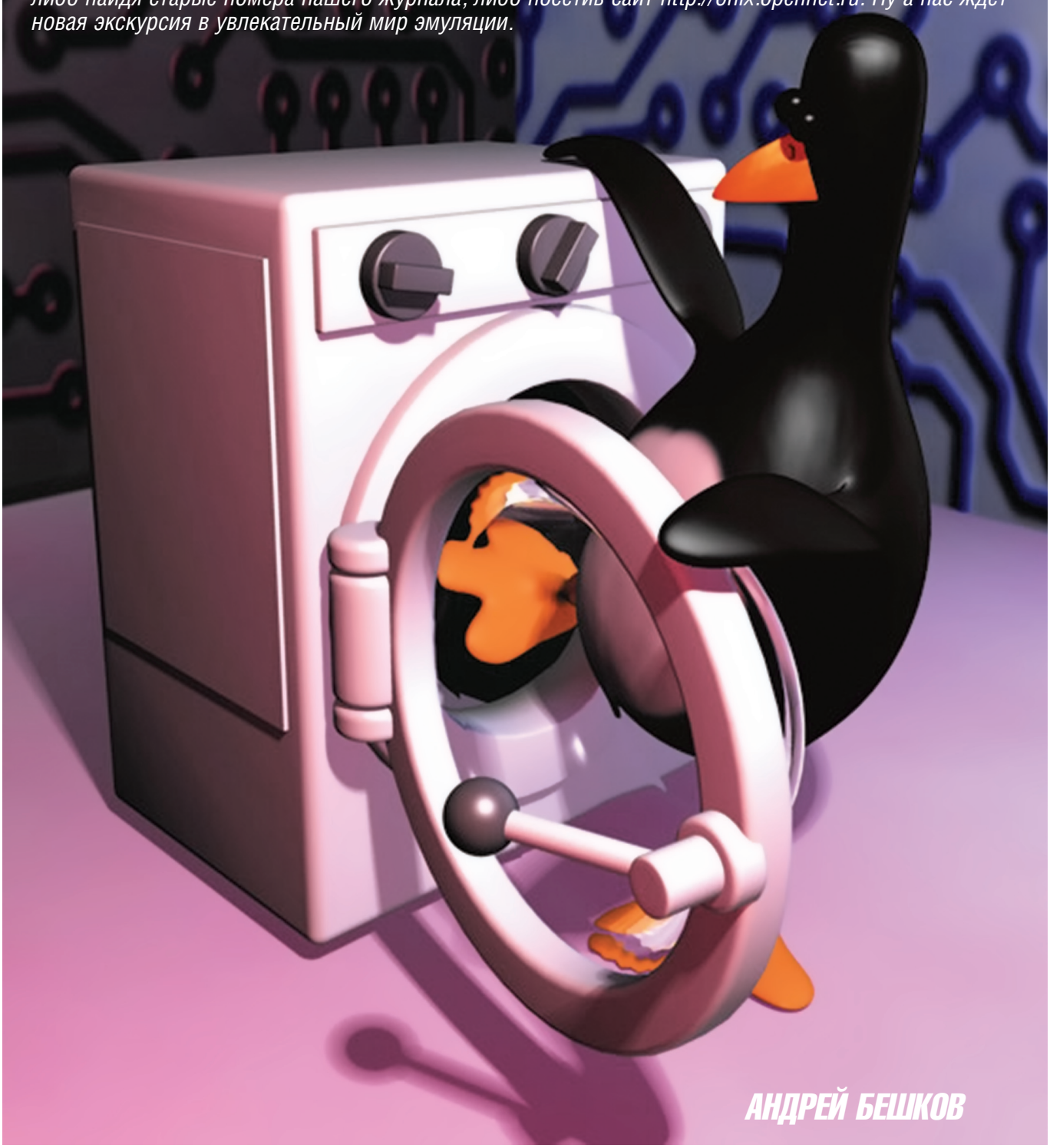
ЗАПУСК WINDOWS-ПРИЛОЖЕНИЙ ПОД LINUX С ПОМОЩЬЮ CROSSOVER OFFICE ЧАСТЬ 3

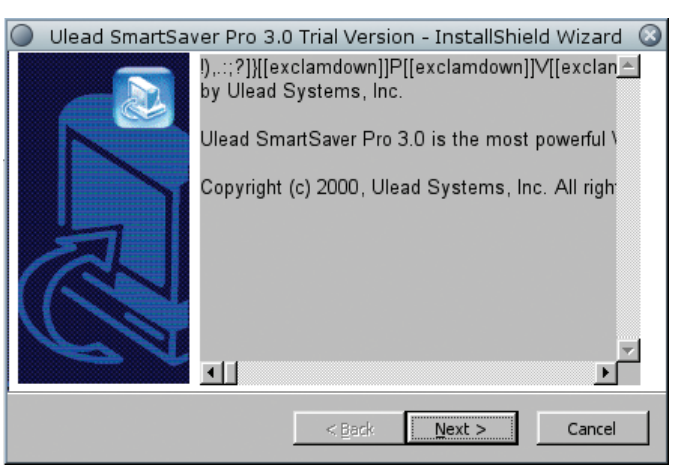
С момента выхода в свет первых двух статей о CrossOver Office прошло не так уж и много времени, и я надеюсь, вы еще не забыли, что мы изучали способы, пользуясь которыми можно довольно просто и удобно работать с Windows-программами под управлением Linux. Если же по каким-либо причинам вы не читали вышеупомянутых статей, то это досадное неудобство можно легко исправить, либо найдя старые номера нашего журнала, либо посетив сайт http://onix.opennet.ru. Ну а нас ждет новая экскурсия в увлекательный мир эмуляции.



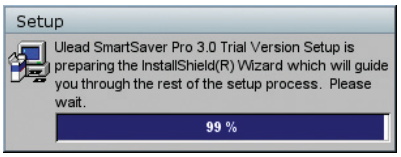
АНДРЕЙ БЕШКОВ

Надеюсь, что третьей статьей мне наконец-то удастся завершить повествование и охватить все вопросы, присланные читателями в ответ на предыдущие публикации по этой теме. Как и обещал, сегодня мы займемся изучением способов поиска неисправностей и скользких моментов, скрытых в Windows-программах и не позволяющих успешно работать со столь необходимыми нам приложениями внутри эмулятора. В дальнейшем я предполагаю, что вы умеете самостоятельно установить и произвести начальную настройку CrossOver Office. Как обычно, готовясь к любым экспериментам внутри эмулятора, лучше всего сделать резервную копию директории /cxoffice, в которой находится наш урезанный вариант Windows. В случае если что-то пойдет не так, как мы ожидаем, у нас всегда будет возможность восстановить рабочую среду простым копированием ее из архива.

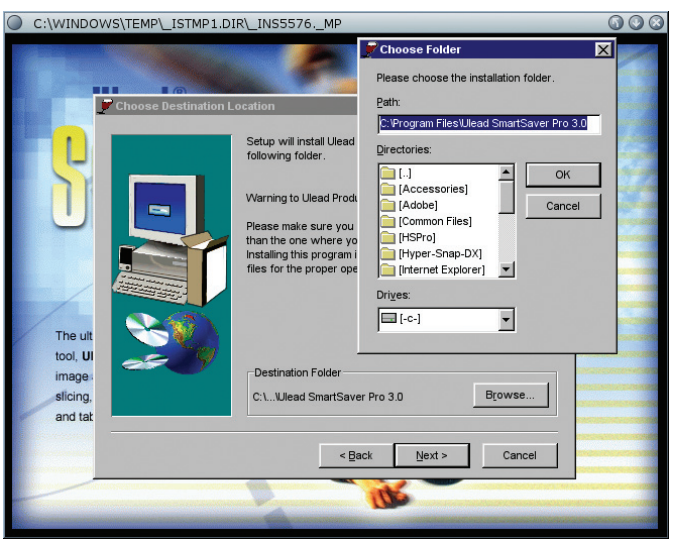
В качестве первого подопытного кролика была выбрана программа Ulead Smart Saver Pro 3.0. Она привлекла мое внимание тем, что умеет очень хорошо оптимизировать изображения, и в то же время в ней присутствуют практически все проблемы, с которыми можно столкнуться при установке Windows-программ. Довольно часто файл, в котором хранится изображение, после обработки этим приложением может похудеть без потери качества на размер от 20 до 80 процентов первоначального объема. Несмотря на все увеличивающееся пропускную способность интернет-каналов, такая оптимизация способна довольно благотворно повлиять на скорость работы любого сайта. Скачиваем дистрибутив пробной версии этого приложения здесь: http://www.ulead.com/ssp/runme.htm. Как обычно, с помощью программы officesetup начинаем новую инсталляцию официально неподдерживаемого программного обеспечения. Первый же появившийся экран программы Ulead Smart Saver Pro заставляет нас предположить что-то неладное: уж очень подозрительно выглядят многочисленные надписи «exclamdown», рассыпавшиеся вперемешку с фигурными скобками по верхней части диалогового окна. Судя по всему, какой-то из служебных скриптов работает неправильно.



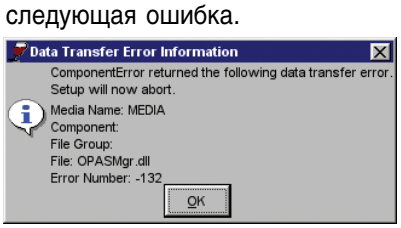
Конечно, это не смертельно, но уже заставляет насторожиться. После перехода к следующему шагу инсталляции по надписям в окне, появляющимся в момент выполнения промежуточных действий, мы видим, что дистрибутив упакован с помощью системы InstallShield.



Обнаружение данного неприятного факта не оставляет нас равнодушными. Поддержка скриптов InstallShield еще не очень хорошо реализована в эмуляторе, поэтому нам придется многие действия выполнять вручную и весьма часто прибегать к помощи шаманского бубна. Впрочем, нас это ни в коей мере не огорчает, потому что пользователи UNIX – люди к трудностям привычные. Так, шаг за шагом мы доходим до момента, когда приходится выбрать, куда именно нужно устанавливать программное обеспечение.



На следующем шаге должно начаться копирование файлов. К сожалению, этого не происходит, и на экране появляется следующая ошибка.



Теперь нужно дождаться, пока программа officesetup не завершит все работающие wine-процессы. Для того чтобы умело пользоваться возможностями отладки, встроенными в wine, нужно изучить немного добавочной теории. Отладочные сообщения могут принадлежать к любой из четырех разновидностей, называемых классами. Принадлежность к тому или иному классу определяется разработчиками wine в зависимости от того, насколько критично для нас выводимое сообщение. Давайте разберемся с каждым из этих пресловутых классов подробнее.

- FIXME – сообщения данного класса сигнализируют о том, что какое-то действие, запрошенное Windows-приложением, не выполнено. Обычно это случается в связи с тем, что реализация нужной нам функции еще не создана в wine, а вместо нее на данный момент используется пустышка, не выполняющая никаких серьезных действий. Такой тип сообщений служит напоминанием разработчикам о необходимости наконец-то закончить разработку того или иного функционала.

- ERR – к подобным сообщениям нужно относиться более серьезно, ведь они показывают наличие критических ошибок во время выполнения кода. А это значит, в процессе работы случилось что-то очень плохое. Чаще всего подробные сведения, разъясняющие, что именно произошло, следуют сразу же за этими сообщениями.
- WARN – предупреждения обычно появляются в тот момент, когда выполняемая функция еще не совершила непоправимых ошибок, но уже не может самостоятельно стопроцентно правильно выполнять все нужные действия. Такие сообщения появляются довольно редко, потому что обычно более или менее правильно написанная функция не должна впадать в панику из-за каких-то мелких нестыковок. В случае ошибок образцовая функция должна вернуть родительской функции код ошибки и предоставить вышестоящим инстанциям право решать, что именно нужно делать.
- TRACE – позволяет предоставить наиболее детализированные отчеты о ходе выполнения того или иного кода. Чаще всего полезен во время первоначальной отладки разработки новых компонентов wine, поэтому по умолчанию отключен.
- MESSAGE – сообщения, предназначенные для конечного пользователя. Так же, как и класс WARN, используется очень редко из-за своей малой полезности в повседневной жизни.

Разобравшись с классами сообщений, давайте перейдем к другому важному понятию. Каждый компонент wine имеет свой канал для вывода отладочной информации. Например, функции, отвечающие за работу с реестром, пишут отладочные сообщения в канал по имени reg, ну а те, кто выполняет действия с файлами, соответственно используют отдельный канал, называемый file. Названия остальных доступных нам каналов выглядят так же просто и понятно. К примеру, функции, занимающиеся загрузкой dll, выводят свои сообщения в канал loaddll. В общей сложности нам доступно 233 канала:

accel	adpcm	advapi	animate	aspi
atom	avicap	avifile	bidi	bitblt
bitmap	cabinet	capi	caret	cdrom
cfgmgr32	class	clipboard	clipping	combo
comboex	comm	commctrl	commdlg	computername
console	crtdll	crypt	curses	cursor
d3d	d3d_shader	d3d_surface	datetime	dc
ddeml	ddraw	ddraw_fps	ddraw_geom	ddraw_tex
debugstr	devenum	dialog	dinput	dll
dma	dmband	dmcompos	dmfile	dmfiledat
dmime	dmloader	dmscript	dmstyle	dmsynth
dmusic	dosfs	dosmem	dplay	dplayx
dpnhpast	driver	dsound	dsound3d	edit
enhmetafile	environ	event	eventlog	exec
file	fixup	font	fps	g711
gdi	global	glu	graphics	header
heap	hook	hotkey	icmp	icon
imagehlp	imagelist	imm	int	int21
int31	io	ipaddress	iphlpapi	jack
joystick	key	keyboard	listbox	listview
loaddll	local	mapi	mci	mcianim
mciavi	mcicda	mcimidi	mciwave	mdi
menu	menubuilder	message	metafile	midi
mmaux	mmio	mmsys	mmtime	module
monthcal	mpeg3	mpr	msacm	msdmo
msg	mshtml	msi	msimg32	msisys
msrle32	msvcrt	msvideo	mswsock	nativefont
netapi32	netbios	nls	nonclient	ntdll
odbc	ole	oledlg	olerelay	opengl
pager	palette	pidl	powermgnt	print
process	profile	progress	propsheet	psapi
psdrv	qcap	quartz	ras	rebar
reg	region	relay	resource	richedit
rundll32	sblaster	scroll	seh	selector
server	setupapi	shdocvw	shell	shlctrl
snmpapi	snoop	sound	static	statusbar
storage	stress	string	syscolor	system
tab	tape	tapi	task	text
thread	thunk	tid	timer	toolbar
toolhelp	tooltips	trackbar	treeview	ttydrv
twain	typelib	uninstaller	updown	urlmon
uxtheme	ver	virtual	vxd	wave
wc_font	win	win32	wineboot	winecfg
wineconsole	wine_d3d	winevdm	wing	winhelp
wininet	winmm	winsock	Winspool	wintab
wintab32	wnet	x11drv	x11settings	xdnd
xrandr	xrender	xvidmode		

Думаю, название каждого из них довольно красноречиво говорит само за себя. Вдобавок ко всем перечисленным есть еще один псевдоканал под названием all, являющийся ссылкой на все каналы сразу. Выбрав его, мы получим абсолютно все отладочные сообщения, создаваемые wine. Количество отладочной информации выводимой в тот или иной канал, может быть довольно большим, если не сказать огромным, поэтому, пользуясь точным указанием интересующих нас каналов и классов сообщений, мы имеем возможность отфильтровать только те фрагменты данных, которые нам реально необходимы. Давайте разберемся, как это делается. Описание типа нужных нам отладочных сообщений чаще всего выглядит довольно просто. К примеру, если мы хотим увидеть сообщения класса WARN для канала, отвечающего за работу с реестром, то должны выполнить следующую команду:

```
$ /opt/cxoffice/bin/wine --debugmsg warn+reg имя программы
```

В случае если класс сообщений не указан, wine считает, что мы хотим читать все классы данного канала. Примером, иллюстрирующим такой подход, может стать следующая команда:

```
$ /opt/cxoffice/bin/wine --debugmsg +reg имя программы
```

Классами сообщения можно управлять не только с помощью знака «+», но и воспользовавшись его антиподом, знаком «-». Следующая приятная возможность, которая может весьма облегчить жизнь во время занятий поиском неисправностей, это то, что каналы сообщений можно перечислять через запятые. Нижеприведенная команда покажет все сообщения, относящиеся к загрузке dll, а из тех, что выводятся при работе с реестром, скроет все, что подпадает под класс ERR.

```
$ /opt/cxoffice/bin/wine --debugmsg +loaddll, ↵
  err-reg имя программы
```

Итак, разобравшись с теорией, приступим к практическим действиям. Нам нужно узнать, что же именно мешает программе установить в систему эту загадочную библиотеку OPASmgr.dll. А так как речь идет об ошибке, связанной с файлом, то, видимо, стоит посмотреть, как приложение обращается со своими файлами. Для этого подаем следующую команду:

```
$ /opt/cxoffice/bin/wine --cx-log logfile.txt ↵
    --debugmsg +file ./Ussp30to.exe
```

Сообщений получается довольно много, поэтому я специально перенаправил их в файл logfile.txt с помощью ключа командной строки --cx-log. В результате получился файл отладки размером в полтора мегабайта. Пройдясь по файлу поиском, обнаруживаем в нем следующие записи:

```
trace:file:CreateFileW L"C:\\WINDOWS\\TEMP\\pft0dc8~tmp\\Programs\\English\\
OPASMgr.dll" GENERIC_READ GENERIC_WRITE FILE_SHARE_READ
FILE_SHARE_WRITE CREATE

trace:file:SetFileAttributesW (L"C:\\WINDOWS\\TEMP\\pft0dc8~tmp\\Programs\\
English\\OPASMgr.dll",fffffffe)

trace:file:SetFileAttributesW (L"C:\\WINDOWS\\TEMP\\pft0dc8~tmp\\Programs\\
English\\OPASMgr.dll",20)
```

Повторяющиеся записи я отфильтровал, но все равно, судя по суете в районе C:\\WINDOWS\\TEMP\\, скрипт Install Shield производит туда распаковку дистрибутива и затем запускает оттуда инсталляцию. Идем далее и обнаруживаем еще более интересные факты.

```
trace:file:CreateFileW L"C:\\Program Files\\Ulead SmartSaver Pro 3.0\\
OPASMgr.dll" GENERIC_READ FILE_SHARE_READ OPEN_EXISTING
attributes 0x80

warn:file:CreateFileW Unable to get full filename from L"C:\\Program Files\\
Ulead SmartSaver Pro 3.0\\ OPASMgr.dll" (GLE 2)

trace:file:CreateFileW L"C:\\Windows\\Temp\\pft0dc8~tmp\\Programs\\English\\
OPASMgr.dll" GENERIC_READ FILE_SHARE_READ OPEN_EXISTING
attributes 0x80

trace:file:CreateFileW returning 0x78

trace:file:GetFileInformationByHandle 0x78

trace:file:CreateFileW L"C:\\Program Files\\Ulead SmartSaver Pro 3.0\\
OPASMgr.dll" GENERIC_WRITE FILE_SHARE_READ CREATE_ALWAYS
attributes 0x80

trace:file:FILE_CreateFile Write access failed for file '/home/newcross/.cxoffice/
dotwine/fake_windows/Program Files/Ulead SmartSaver Pro 3.0/OPASMgr.dll

warn:file:FILE_CreateFile Unable to create file '/home/newcross/.cxoffice/
dotwine/fake_windows/Program Files/Ulead SmartSaver Pro 3.0/OPASMgr.dll'
(GLE 5)

trace:file:CreateFileW returning 0xffffffff

trace:file:CreateFileW L"C:\\Program Files\\Ulead SmartSaver Pro 3.0\\
OPASMgr.dll" GENERIC_READ FILE_SHARE_READ OPEN_EXISTING
attributes 0x80

warn:file:CreateFileW Unable to get full filename from L"C:\\Program Files\\
Ulead SmartSaver Pro 3.0\\ OPASMgr.dll" (GLE 2)
```

Судя по всему, инсталлятор не может получить доступа на запись в собственноручно созданную директорию C:\\Program Files\\Ulead SmartSaver Pro 3.0\\, а возможно, ему не удается внести нужные данные в файл C:\\Program Files\\Ulead SmartSaver Pro 3.0\\OPASMgr.dll. Посмотрев на вышеуказанную директорию, понимаем, что с правами у нас все вроде бы в порядке.

```
drwx------  2 tigrisha tigrisha 4096 Map 15 22:02 Ulead SmartSaver Pro 3.0/
```

Весьма настораживает активность внутри директории C:\\Windows\\Temp\\, поэтому запускаем инсталляцию вновь, но уже без отладки. Дойдя до момента, когда нужно указать, куда устанавливать программу, останавливаемся и смотрим, что у нас появилось нового в интересующей нас директории. Там действительно есть на что поглядеть, особенно обращают на себя внимание папки, иерархия которых отображена на следующей картинке.

```
Windows
└─ Temp
   ├─ _ISTMP1.DIR
   │  └─ _ISTMP0.DIR
   └─ pft82b0~tmp
      └─ Programs
         └─ English
            ├─ Aps
            ├─ Audio
            ├─ Fio
            ├─ Mask
            ├─ Opas
            │  ├─ 27
            │  ├─ 28
            │  └─ 64
            ├─ Samples
            │  ├─ audio
            │  └─ images
            └─ Upi
```

Пройдясь по этим папкам, мы обнаруживаем, что внутри них хранится уже распакованный и готовый к установке дистрибутив. Копируем его в какое-либо безопасное место и прерываем ожидающую нашего ответа программу инсталляции. Затем уже запускаем программу Setup.exe из только что распакованного дистрибутива. Теперь вся установка, освобожденная от оков InstallShield, проходит на ура, но радоваться пока рано. Попробовав запустить свежеустановленную программу с помощью файла Usspro.exe, получаем от ворот поворот. Приложение, не открывая ни одного окна и не показывая никаких ошибок, молчаливо вываливается обратно в консоль. Вот тут-то нам и пригодятся отладочные каналы loaddll и file. Проблемы с загрузкой тех или иных библиотек – довольно частая причина, по которой приложения отказываются работать.

```
$ /opt/cxoffice/bin/wine --debugmsg +file ./Usspro.exe
```

```
err:module:import_dll Module (file) u32Comm.dll (which is needed by
C:\Program Files\Ulead SmartSaver Pro 3.0\UssAbout.dll) not found

err:win32:PE_LoadLibraryExA can't load C:\Program Files\
Ulead SmartSaver Pro 3.0\UssAbout.dll

err:module:import_dll Loading module (file) UssAbout.dll (which is needed by
C:\Program Files\Ulead SmartSaver Pro 3.0\Usspro.exe) failed (error c0000017)
```

Судя по первой строке, программа не может найти файл динамически загружаемой библиотеки u32Comm.dll, на которую в свою очередь ссылается UssAbout.dll. При попытке приложения загрузить динамические библиотеки wine ищет их в следующих местах:

- папка, откуда программа была запущена (где лежит ее выполняемый файл);
- текущая папка;
- папка C:\Windows\System\;
- папка C:\Windows\;
- все остальные папки, указанные в переменной окружения PATH.

В случае если программе нужна какая-либо специфическая DLL, поставлявшаяся вместе с Windows, ее можно перенести вручную либо с компьютера, работающего под этой операционной системой, либо найти ее в Интернете. Ну а если с этими двумя вариантами не повезло, то всегда остается возможность с помощью ути-

литы cabextract вытащить необходимые файлы из cab-архивов, хранящихся на CD-ROM с дистрибутивом Windows. Довольно быстро обнаруживаем недостающую библиотеку в дистрибутиве программы и кладем ее в домашнюю директорию приложения. После этого можно снова попробовать запустить его.

На этот раз нам повезло чуть больше, появилась заставка, рассказывающая о том, что программа защищена с помощью системы VBOX, и что мы можем пользоваться ею условно бесплатно в течение следующих 15 дней. Наличие той или иной защиты в программе обычно довольно сильно усложняет процесс работы с таким приложением. К сожалению, приложение все еще не готово к нормальной работе. Нажав на кнопку «Try», получаем огромный список предупреждений и критическую ошибку, вызывающую немедленное падение программы. Снова запускаем наше многострадальное приложение, только теперь нас интересуют данные из отладочного канала loaddll.

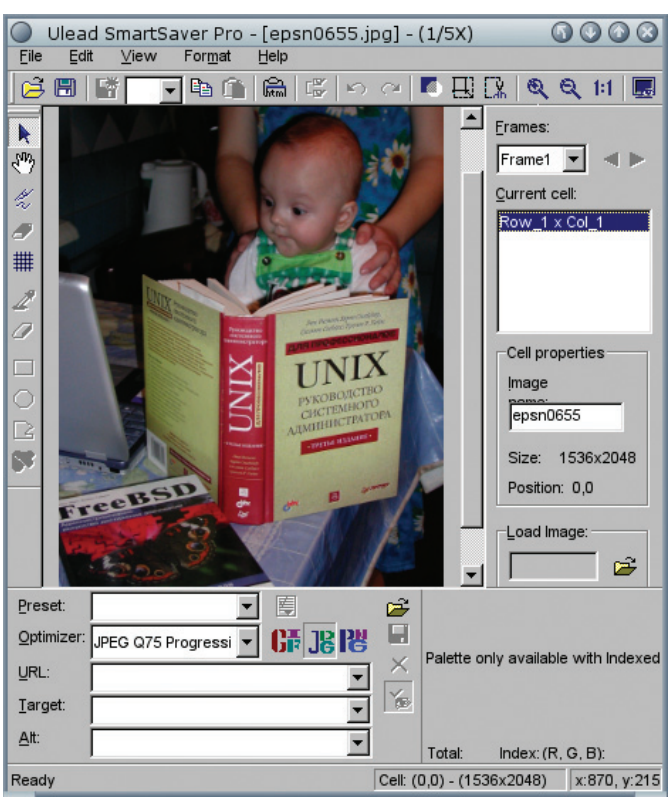
```
$ /opt/cxoffice/bin/wine --debugmsg +loaddll ./Usspro.exe
```

```
trace:loaddll:load_dll Loaded module 'C:\Windows\System\ADVAPI32.DLL' : builtin
trace:loaddll:load_dll Loaded module 'C:\WINDOWS\SYSTEM\gdi32.dll' : builtin
trace:loaddll:load_dll Loaded module 'C:\WINDOWS\SYSTEM\USER32.dll' : builtin
trace:loaddll:load_dll Loaded module 'C:\Windows\System\ole32.dll' : native
trace:loaddll:load_dll Loaded module 'C:\Windows\System\SHLWAPI.DLL' : native
trace:loaddll:load_dll Loaded module 'C:\Windows\System\COMCTL32.DLL' : builtin
trace:loaddll:load_dll Loaded module 'C:\Windows\System\SHELL32.DLL' : builtin
trace:loaddll:load_dll Loaded module 'C:\Program Files\Ulead SmartSaver Pro 3.0\
U32base.dll' : native
trace:loaddll:load_dll Loaded module 'C:\Program Files\Ulead SmartSaver Pro 3.0\
U32sn.dll' : native
trace:loaddll:load_dll Loaded module 'C:\Program Files\Ulead SmartSaver Pro 3.0\
U32prod.dll' : native
trace:loaddll:load_dll Loaded module 'C:\Program Files\Ulead SmartSaver Pro 3.0\
U32cfg.dll' : native
trace:loaddll:load_dll Loaded module 'C:\Program Files\Ulead SmartSaver Pro 3.0\
UssAbout.dll' : native
err:module:import_dll No implementation for SHLWAPI.dll.AssocIsDangerous
imported from C:\Windows\System\SHDOCVW.DLL, setting to 0xdeadbeef
trace:loaddll:load_dll Loaded module 'C:\Windows\System\SHDOCVW.DLL' : native
trace:loaddll:load_dll Loaded module 'C:\Windows\System\MSOSS.DLL' : native
wine: Unhandled exception (thread 000f), starting debugger...
trace:loaddll:load_dll Loaded module 'C:\Windows\System\ADVAPI32.DLL' : builtin
trace:loaddll:MODULE_LoadModule16 Loaded module 'krnl386.exe' : builtin
trace:loaddll:MODULE_LoadModule16 Loaded module 'system.drv' : builtin
WineDbg starting on pid e
No debug information in ELF '/home/newcross/cxoffice/bin/wineloader' ((nil))
Breakpoint 1 at 0x4000a090
No debug information in ELF '/home/newcross/cxoffice/lib/libntdll.dll.so'
(0x40012000)
No debug information in ELF '/home/newcross/cxoffice/lib/libwine.so.1'
(0x400eb000)
No debug information in ELF '/home/newcross/cxoffice/lib/libwine_unicode.so.1'
(0x40104000)
.....
```

На приведенной краткой выдержке из протокола явно видно, что большинство DLL работает в режиме native, то есть wine не пытается подменять своими собственными реализациями родные библиотеки. Судя по строкам, которые я выделил красным цветом, у нас происходит конфликт версий между библиотеками SHDOCVW.DLL и SHLWAPI.DLL. Вторая библиотека требует от первой правильной реализации импортируемой функции AssocIs Dangerous, но, судя по всему, выполнить эти требования SHLWAPI.DLL не способна. Для выхода из столь неприятного положения мы можем принудительно указать wine, что при загрузке SHDOCVW.DLL и SHLWAPI.DLL нужно использовать не родные версии этих библиотек, а встроенные (builtin) в wine. Поэтому нам придется впредь запускать приложение следующей командой:

```
$ /opt/cxoffice/bin/wine --dll shdocvw,shlwapi=b ./Usspro.exe
```

По идее, можно было обойтись только подменой SHDOCVW.DLL, но, к сожалению, это не всегда стабильно работает. Как видите, приложение отлично функционирует. Убедиться в этом можно, посмотрев на следующий снимок экрана.



Теперь программу придется запускать указанной выше командной строкой, что, согласитесь, немного неудобно. Ну а мы, как истинные сибариты, хотим избавить себя от запоминания и постоянного набора с клавиатуры этих опций, поэтому открываем конфигурационный файл $HOME/fake_windows/config и ищем в нем вот такую строчку: # [/wineconf], обозначающую конец главной секции. Найдя ее, вставляем в любое приглянувшееся место перед этой строкой следующее объявление, описывающее необходимое нам переопределение порядка загрузки библиотек для приложения Usspro.exe.

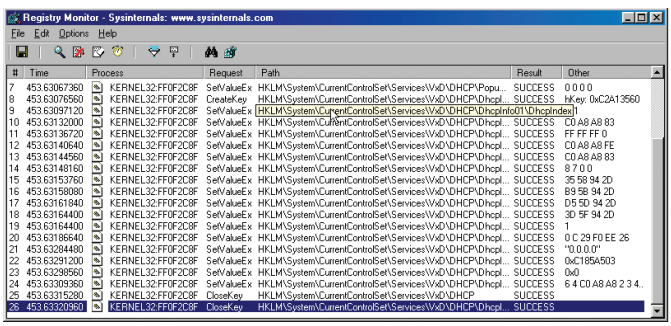
```
[AppDefaults\\Usspro.exe\\DllOverrides]
; Ulead Smart Saver Pro 3.0
"shdocvw" = "builtin"
"shlwapi" = "builtin
```

После сохранения файла конфигурации можно будет удобнее и спокойнее работать с приложением без страха забыть какую-либо опцию или уронить другие программы, которым для правильного функционирования нужно обязательно загружать родные варианты вышеназванных DLL.

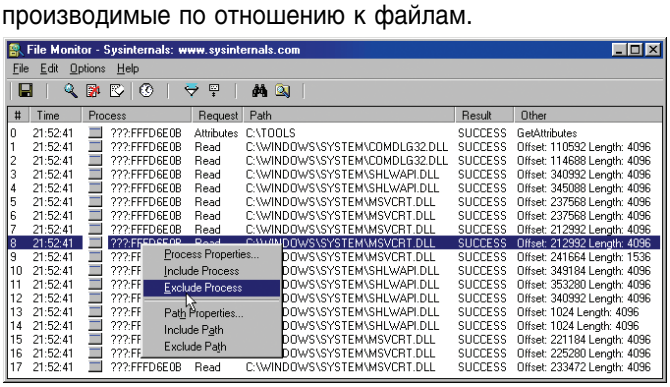
Закончив бороться с этой программой, мы многому научились, но пока что не время останавливается. Поэтому сейчас нужно заняться установкой программы Ultra Edit версии 10.10b, которую можно скачать, перейдя по следующему адресу: http://www.ultraedit.com/downloads/. Данное приложение выбрано в качестве объекта эксперимента по двум причинам. Во-первых, потому, что оно является, с моей точки зрения, одним из самых удобных редакторов для разработчика из всех когда-либо встречаемых мной под Windows. Вторая причина состоит в том, что из-за очень сложного кода инсталлятора программу тяжело установить и успешно эксплуатировать под управлением эмулятора. А нам это как нельзя кстати. Позарез нужны именно такие тяжелые пациенты, потому что, заставив эту программу работать, мы сможем изучить несколько новых продвинутых приемов. Это в свою очередь позволит существенно повысить успешность наших попыток по переносу Windows-приложений в среду эмуляции. Добавочный опыт выживания в сложных условиях будет для нас очень кстати. Пробуем запустить инсталляцию и убеждаемся в том, что дела идут хуже некуда. Программа, нарисовав на экране заставку, падает через полсекунды, буквально засыпав экран ворохом ошибок. Опробовав все приемы, изученные нами в борьбе с предыдущим противником, понимаем, что проблема ни на йоту не сдвинулась с мертвой точки, а все приобретенные знания ничем не помогают.

Пришло время бросить в бой танковые дивизии и задавить противника техническим преимуществом. Для создания полигона нам понадобится либо отдельный компьютер с работающей Windows 98, либо система полной эмуляции с запущенной внутри нее полноценной версией нужной нам операционной системы. В качестве таких эмулирующих контейнеров можно использовать VMWare Workstation, Virtual PC или Win4Lin. Я выбрал VMWare как наиболее привычный для меня инструмент, ну а вы можете использовать любой из вышеперечисленных вариантов, самый подходящий под ваши вкусовые предпочтения. Главное, чтобы у нас была нормальная рабочая версия Windows, под управлением которой мы сможем проинсталлировать все необходимые инструменты. Первым делом устанавливаем программу Filemon, которая позволяет протоколировать все обращения к файлам. Затем делаем то же самое с программой Regmon, помогающей следить за всеми манипуляциями с реестром. Автор обоих вышеописанных инструментов Mark Russinovich. Они лежат в свободном доступе на сайте: http://www.sysinternals.com/win9x/98utilities.shtml. Установка проста, как три копейки. Создайте на жестком диске директорию и распакуйте в нее содержимое архивов. Думаю, инсталляция этого комплекса ни у кого не вызовет затруднений.

Пользуясь только что полученными инструментами, мы сможем записать все действия, которые подопытная программа выполняет во время своей установки. А затем повторить их вручную под управлением CrossOver, тем самым освобождая себя от необходимости пользоваться фирменным инсталлятором. Запустив Regmon, должны увидеть на экране похожую картинку.

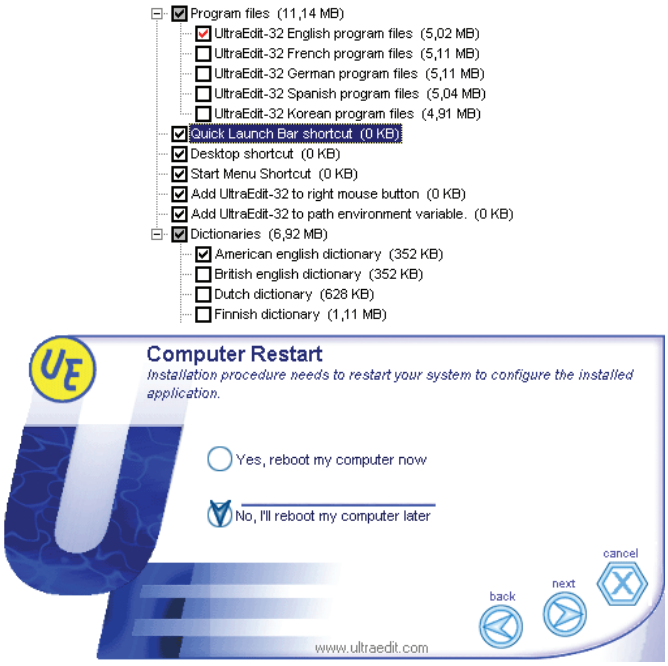


Как вы могли убедиться, интерфейс приложения прост и интуитивно понятен. При каждом обращении любого процесса к реестру программа записывает время, когда случилось событие, имя процесса, тип действия, ключ реестра, с которым нужно взаимодействовать, код возврата и данные, полученные в результате выполнения действия. Интерфейс утилиты Regmon выглядит точно так же, единственное отличие в том, что он показывает действия, производимые по отношению к файлам.



Обычно в системе работает очень много процессов, занимающихся своими собственными делами, дабы они не мешали нам и не мусорили в протокол, мы должны сузить поле зрения запущенных утилит. Для этого клавишей щелкаем на неугодном процессе и в ниспадающем меню выбираем пункт «Exclude Process». Исключая один процесс за другим, мы должны добиться, чтобы протоколировались только действия процесса, занимающегося инсталляцией. Запустив установку UltraEdit, выбираем, какой тип инсталляции провести.

Затем решаем, какие из компонентов нам действительно необходимы, а от каких можно безболезненно отказаться. После того как установка успешно завершится, на вопрос, нужно ли перезагрузить операционную систему, непременно отвечаем «нет».



Делаем мы так по двум причинам. Во-первых, данные слежения, собранные с помощью Regmon и Filemon, еще не сохранены на жесткий диск. Во-вторых, после перезагрузки состояние системы может сильно измениться, а некоторые файлы могут быть удалены. Хотелось бы как можно точнее имитировать внутри эмулятора всю последовательность событий, происходящих во время установки и перезагрузки.

Сохраняем протоколы работы с файловой системой и ключами реестра в файлы uedit_file_inst.log и uedit_reg_inst.log. После этого можно посмотреть, что, собственно, в них записалось. Здесь я приведу только краткие выдержки, собранные из наиболее характерных записей.

Содержимое файла uedit_file_inst.log:

```
64    22:23:01    ???:FFFC5C7B    Attributes    C:\HACK\UESETUP.EXE
SUCCESS GetAttributes
65    22:23:01    ???:FFFC5C7B    Directory    C:\HACK\UESETUP.EXE
SUCCESS QUERY
66    22:23:01    ???:FFFC5C7B    Open    C:\HACK\UESETUP.EXE
SUCCESS OPENEXISTING READONLY DENYWRITE
67    22:23:01    ???:FFFC5C7B    Read    C:\HACK\UESETUP.EXE
SUCCESS Offset: 0 Length: 64
68    22:23:01    ???:FFFC5C7B    Seek    C:\HACK\UESETUP.EXE
SUCCESS Beginning Offset: 256 / New offset: 256
.....
112    22:23:01    Uesetup:FFFC5C7B    FindOpen
C:\WINDOWS\WIN.INI    SUCCESS WIN.INI
113    22:23:01    Uesetup:FFFC5C7B    FindClose
C:\WINDOWS\WIN.INI    SUCCESS
.....
119    22:23:01    Uesetup:FFFC5C7B    Read    C:\WINDOWS\WIN.INI
SUCCESS Offset: 0 Length: 7491
120    22:23:01    Uesetup:FFFC5C7B    Close    C:\WINDOWS\WIN.INI
SUCCESS CLOSE_FINAL
.....
711    22:23:01    Uesetup:FFFC5C7B    Write    C:\WINDOWS\TEMP\
1C27LT8C\UNPACK.DLL    SUCCESS Offset: 0 Length: 35328
712    22:23:01    Uesetup:FFFC5C7B    Open    C:\WINDOWS\WIN.INI
SUCCESS OPENEXISTING READWRITE DENYWRITE
713    22:23:01    Uesetup:FFFC5C7B    Ioctl    C:    SUCCESS Subfunction:
08h
714    22:23:01    Uesetup:FFFC5C7B    Seek    C:\WINDOWS\WIN.INI
SUCCESS Beginning Offset: 0 / New offset: 0
```

Самыми интересными для нас являются строки, в которых упоминается операция Write, ведь мы хотим узнать, какие файлы появились и какие изменились в результате установки. Для удобства я пометил пример такой строки красным цветом. Теперь давайте посмотрим на отрывки из файла протокола работы с реестром.

Содержимое файла uedit_reg_inst.log:

```
16    40.46406720    Uesetup:FFFC5C7B    OpenKey HKCU\Control Panel\
Desktop    SUCCESS hKey: 0xC2A14FD0
17    40.46411120    Uesetup:FFFC5C7B    QueryValueEx    HKCU\Control Panel\
Desktop\SmoothScroll NOTFOUND
18    40.46414480    Uesetup:FFFC5C7B    CloseKey    HKCU\Control Panel\
Desktop    SUCCESS
19    40.46420480    Uesetup:FFFC5C7B    OpenKey HKCU\Control Panel\Mouse
NOTFOUND
.....
543    102.20113120    Uesetup:FFFC5C7B    CreateKey    HKLM\SOFTWARE\
Microsoft\Windows\CurrentVersion\SharedDLLs    SUCCESS hKey: 0xC2A15080
544    102.20122880    Uesetup:FFFC5C7B    SetValueEx    HKLM\SOFTWARE\
Microsoft\Windows\CurrentVersion\SharedDLLs\C:\Program Files\UltraEdit\
UninstallExe    SUCCESS 0x1
545    44.74403920    Uesetup:FFFC5C7B    EnumValue    HKLM\SYSTEM\
CurrentControlSet\Control\NIs\Locale    SUCCESS 0000040D: ""
```

Как и в предыдущем примере, интересующие нас строки помечены красным. Надеюсь, вы уже догадались, что нас интересуют операции CreateKey и SetValueEx, отвечающие за создание новых ключей и установку их значений. Впрочем, просмотрев файл, я пришел к выводу, что операция SetValueEx, присваивающая ключу значение, всегда идет сразу же за CreateKey, поэтому можно облегчить нашу задачу, выбирая данные только по этому признаку. Поразмыслив еще немного, я пришел к выводу, что такое поведение довольно естественно, ведь программе инсталляции нет смысла создавать пустые ключи.

В связи с тем, что размер первого файла 1.2 Мб, а второго 283 Кб, думаю, все понимают, что обрабатывать вручную их не только затруднительно, но и весьма неприятно. Поэтому мы поступим в соответствии с фразой «Никогда не доверяй человеку работу, которую может выполнить скрипт». По моему мнению, UNIX является лучшей платформой для выполнения задачи фильтрации текстовых данных. Так происходит потому, что благодаря возможности связывать команды в цепочку с перенаправлением результатов работы одной команды на стандартный ввод другой мы можем без труда создавать довольно сложные фильтры.

Итак, давайте нарисуем план работы скрипта, фильтрующего данные о действиях с файлами. Сначала нужно выбрать все строки, в которых тип операции равен Write, затем отфильтровать только те, где статус выполняемой операции равен SUCCES. После этого вырезать пятое поле строки, содержащее имя изменяемого файла. В связи с тем, что во время первоначальной распаковки дистрибутива происходит очень много операций записи в папку TEMP, нам нужно отбросить все строки, содержащие такие фрагменты. Делаем мы так потому, что после инсталляции программа стирает все временные файлы из папки TEMP, соответственно нам там тоже нечего делать. Полученный список файлов подвергается сортировке с исключением повторяющихся строк и сохраняется в файл uedit_file_inst_selected.log. Все вышеописанные действия выполняются следующей командой:

```
$ cat uedit_file_inst.log | grep Write | grep SUCCESS | cut ↵
    -f5> | grep -v "TEMP" | sort -u > uedit_file_inst_selected.log
```

В результате получаем файл размером в 1.4 Кб с вот таким содержимым:

```
C:\1G27M3QF.BAT
C:\PROGRAM FILES\ULTRAEDIT\AUTOCORR.TLX
C:\PROGRAM FILES\ULTRAEDIT\CHANGES.TXT
C:\PROGRAM FILES\ULTRAEDIT\CONDCORR.TLX
C:\PROGRAM FILES\ULTRAEDIT\HTMLTIDY.DLL
C:\PROGRAM FILES\ULTRAEDIT\HTML.TLX
C:\PROGRAM FILES\ULTRAEDIT\ORDER.TXT
C:\PROGRAM FILES\ULTRAEDIT\READ.ME
C:\PROGRAM FILES\ULTRAEDIT\SFTPDLL.DLL
C:\PROGRAM FILES\ULTRAEDIT\SSCE4332.DLL
C:\PROGRAM FILES\ULTRAEDIT\SSCEAM1.CLX
C:\PROGRAM FILES\ULTRAEDIT\SSCEAM.TLX
C:\PROGRAM FILES\ULTRAEDIT\TAGLIST.TXT
C:\PROGRAM FILES\ULTRAEDIT\UCL.CHM
C:\PROGRAM FILES\ULTRAEDIT\UCL.EXE
C:\PROGRAM FILES\ULTRAEDIT\UCRES.DLL
C:\PROGRAM FILES\ULTRAEDIT\UE32CTMN.DLL
C:\PROGRAM FILES\ULTRAEDIT\UEDIT32.EXE
C:\PROGRAM FILES\ULTRAEDIT\UEDIT32.HLP
C:\PROGRAM FILES\ULTRAEDIT\UEDOS32.EXE
C:\PROGRAM FILES\ULTRAEDIT\UEINSTALL.log
C:\PROGRAM FILES\ULTRAEDIT\UEINSTALL.SSS
C:\PROGRAM FILES\ULTRAEDIT\UERES.DLL
C:\PROGRAM FILES\ULTRAEDIT\UNINSTALL.EXE
C:\PROGRAM FILES\ULTRAEDIT\WORDFILE.TXT
C:\WINDOWS\UEDIT32
C:\WINDOWS\WIN386.SWP
C:\WINDOWS\РАБОЧИЙ СТОЛ\ULTRAEDIT-32.LNK
C:\WINDOWS\ГЛАВНОЕ МЕНЮ\ULTRAEDIT-32.LNK
C:\WINDOWS\ГЛАВНОЕ МЕНЮ\ПРОГРАММЫ\ULTRAEDIT\ULTRAEDIT-32
HELP.LNK
C:\WINDOWS\ГЛАВНОЕ МЕНЮ\ПРОГРАММЫ\ULTRAEDIT\ULTRAEDIT-32
ORDER FORM.LNK
C:\WINDOWS\ГЛАВНОЕ МЕНЮ\ПРОГРАММЫ\ULTRAEDIT\ULTRAEDIT-32
READ ME.LNK
C:\WINDOWS\ГЛАВНОЕ МЕНЮ\ПРОГРАММЫ\ULTRAEDIT\ULTRAEDIT-32
TEXT EDITOR.LNK
C:\WINDOWS\ГЛАВНОЕ МЕНЮ\ПРОГРАММЫ\ULTRAEDIT\ULTRAEDIT-32
UNINSTALL.LNK
```

Складываем в отдельную папку файлы, указанные в списке. Иерархию директорий, в которой эти файлы изначально находились, лучше всего воссоздать в нашей временной папке. Так будет проще разложить файлы по нужным местам в эмуляторе. Впрочем, файлы WIN386.SWP и все, что связано с рабочим столом и главным меню, можно не копировать.

Настало время приступить к выполнению действий над файлом, в котором хранится протокол работы с реестром. Процедура фильтрации отличается от предыдущего примера только тем, что мы выбираем строки, в которых встречается цепочка символов SetValue.

```
$ cat uedit_reg_inst.log | grep SUCCESS | grep "SetValue" | ↵
    cut -f5 | sort -u > tmp.log
```

В качестве награды за труды и проявленную сообразительность получаем файл tmp.log размером в 2.9 Кб.

```
0xC2A13710\SavedLegacySettings
HKCC\Software\Microsoft\windows\CurrentVersion\Internet Settings\ProxyEnable
HKCR\CLSID\{b5eedee0-c06e-11cf-8c56-444553540000}\InProcServer32
HKCR\CLSID\{b5eedee0-c06e-11cf-8c56-444553540000}\InProcServer32\
ThreadingModel
HKCR\*\shellex\ContextMenuHandlers\UltraEdit-32
HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\
GlobalUserOffline
HKCU\Software\Microsoft\windows\CurrentVersion\Internet Settings\MigrateProxy
HKCU\Software\Microsoft\windows\CurrentVersion\Internet Settings\ProxyEnable
HKCU\Software\Microsoft\windows\CurrentVersion\Internet Settings\ProxyServer
HKLM\Software\IDM Computer Solutions, Inc.\UltraEdit-32\GroupName
HKLM\Software\Microsoft\Windows\CurrentVersion\App Paths\UEDIT32.exe
HKLM\Software\Microsoft\Windows\CurrentVersion\App Paths\UEDIT32.exe\Path
HKLM\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Cache\Paths\
Directory
HKLM\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Cache\Paths\
Path1\CacheLimit
HKLM\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Cache\Paths\
Path1\CachePath
HKLM\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Cache\Paths\
Path2\CacheLimit
HKLM\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Cache\Paths\
Path2\CachePath
HKLM\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Cache\Paths\
Path3\CacheLimit
HKLM\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Cache\Paths\
Path3\CachePath
HKLM\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Cache\Paths\
Path4\CacheLimit
HKLM\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Cache\Paths\
Path4\CachePath
HKLM\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Cache\Paths\
Paths
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\SharedDLLs\
C:\Program Files\UltraEdit\Uninstall.exe
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\
{43B6667D-7520-4186-B05B-F5C0494C495D}\DisplayName
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\
{43B6667D-7520-4186-B05B-F5C0494C495D}\DisplayVersion
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\
{43B6667D-7520-4186-B05B-F5C0494C495D}\InstallDate
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\
{43B6667D-7520-4186-B05B-F5C0494C495D}\InstallLocation
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\
{43B6667D-7520-4186-B05B-F5C0494C495D}\InstallSource
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\
{43B6667D-7520-4186-B05B-F5C0494C495D}\InstallSourceFile
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\
{43B6667D-7520-4186-B05B-F5C0494C495D}\Publisher
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\
{43B6667D-7520-4186-B05B-F5C0494C495D}\SilentSettings
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\
{43B6667D-7520-4186-B05B-F5C0494C495D}\UninstallString
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\
{43B6667D-7520-4186-B05B-F5C0494C495D}\URLInfoAbout
C:\WINDOWS\ГЛАВНОЕ МЕНЮ\ПРОГРАММЫ\ULTRAEDIT\
ULTRAEDIT-32 ORDER FORM.LNK
C:\WINDOWS\ГЛАВНОЕ МЕНЮ\ПРОГРАММЫ\ULTRAEDIT\
ULTRAEDIT-32 READ ME.LNK
C:\WINDOWS\ГЛАВНОЕ МЕНЮ\ПРОГРАММЫ\ULTRAEDIT\
ULTRAEDIT-32 TEXT EDITOR.LNK
C:\WINDOWS\ГЛАВНОЕ МЕНЮ\ПРОГРАММЫ\ULTRAEDIT\
ULTRAEDIT-32 UNINSTALL.LNK
```

Как вы теперь можете убедиться, всю рутинную работу удалось выполнить очень легко. Именно в таких задачах проявляется скромное очарование UNIX.

Первая строка этого файла слегка сбила меня с толку. Поискав в реестре, я так и не смог найти разделов и ключей с именем 0xC2A13710. Но, с другой стороны, такой раздел должен быть, ведь мы отбирали записи только о тех операциях с реестром, которые были действительно выполнены и возвратили код SUCCESS. Немного подумав, я стал искать с помощью программы regedit подразделы с именем SavedLegacySettings. Такая цепочка символов встречалась лишь в следующих подразделах:

```
HKEY_USERS\.DEFAULT\SOFTWARE\Microsoft\Windows\CurrentVersion\
Internet Settings\Connections
HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\
Internet Settings\Connections
```

Добавив их в файл вместо символов 0xC2A13710, я решил, что можно приступать к экспорту данных из реестра. По идее можно было бы еще сильнее уменьшить этот файл, сведя все ключи к минимальному общему набору символов. К примеру, следующие ключи:

```
HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\
GlobalUserOffline
HKCU\Software\Microsoft\windows\CurrentVersion\Internet Settings\MigrateProxy
HKCU\Software\Microsoft\windows\CurrentVersion\Internet Settings\ProxyEnable
HKCU\Software\Microsoft\windows\CurrentVersion\Internet Settings\ProxyServer
```

могут быть заменены одной строкой:

```
HKCU\Software\Microsoft\windows\CurrentVersion\Internet Settings
```

Подобным же образом можно поступить и с многими другими записями. При экспортировании любой ветки с помощью regedit все вложенные в нее значения обязательно будут сохранены в файл.

Первая проблема заключается в том, что Regmon записывает имена ключей в краткой форме, но regedit не понимает такой нотации. Соответственно мы, получив имя ключа реестра, должны приводить его к стандартному виду, заменяя сокращенные названия на полные в соответствии со следующей таблицей.

HKCR	HKEY_CLASSES_ROOT
HKCU	HKEY_CURRENT_USER
HKLM	HKEY_LOCAL_MACHINE
HKU	HKEY_USERS
HKCC	HKEY_CURRENT_CONFIG
HKDD	HKEY_DYN_DATA

Сделать это можно следующей командой, использующей возможности строкового редактора sed:

```
$ cat temp.log | sed -e 's/HKCR/HKEY_CLASSES_ROOT/' -e ↵
    's/HKCU/HKEY_CURRENT_USER/' -e ↵
    's/HKLM/HKEY_LOCAL_MACHINE/' -e 's/HKU/HKEY_USER S/' ↵
    -e 's/HKCC/HKEY_CURRENT_CONFIG/' -e ↵
    's/HKDD/HKEY_DYN_DATA/' > uedit_reg_inst_selected.log
```

Переносим полученный файл uedit_reg_inst_selected.log обратно под Windows 98. Затем с помощью regedit проходим по веткам, перечисленным в списке, и проводим экспортирование каждой из них в файл. Полученные файлы переносим на UNIX и соединяем в один с помощью вот такой простой команды:

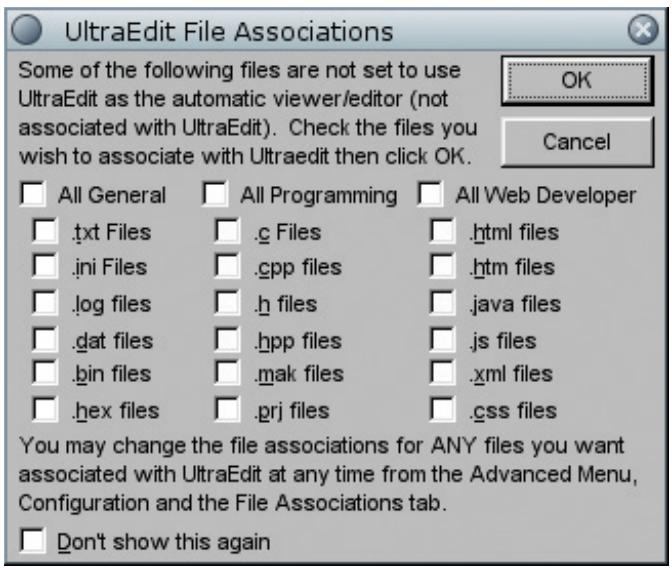
```
$ echo "REGEDIT4" > final.reg; cat *.reg | grep -v ↵
    "REGEDIT4" >> final.reg
```

Следующая проблема, которая нас поджидает на пути, состоит в том, что каждый reg-файл обязательно должен начинаться строкой REGEDIT4. Поэтому, при слиянии всех файлов в один, наш результирующий файл был буквально напичкан этой надписью. Пришлось удалить все лишние строки и оставить только самую первую. Закончив с этим, можно приступить к импортированию полученных данных в реестр Windows, работающий под управлением CrossOver Office.

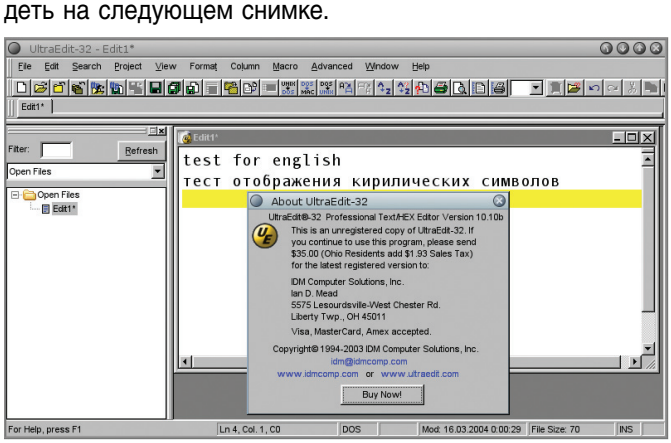
```
$ /opt/cxoffice/bin/regedit final.reg
```

Нам нужно вручную воссоздать всю иерархию директорий, находящихся в C:\PROGRAM FILES\ULTRAEDIT\. После этого можно приступать к раскладыванию бинарных файлов, собранных нами под Windows 98 по соответствующим директориям. Закончив с этим, обязательно выполняем перезапуск нашей виртуальной Windows системы с помощью программы cxreboot.

Самое время проверить результаты нашего долгого труда. Переходим в домашнюю директорию Ultraedit и пытаемся запустить файл uedit32.exe. Как обычно, при первом запуске получаем экран с просьбой назначить расширения, связываемые с UltraEdit.



И затем нашему взору предстают и все остальные рабочие окна редактора. Как это выглядит, вы можете увидеть на следующем снимке.



Я надеюсь, теперь многие из читателей убедились, что мы с успехом выполнили то, что в начале пути казалось совершенно неосуществимым, нужно было лишь приложить немного труда и научиться нескольким удачным приемам. Стоит признать, что методики переноса приложений, описанные в данной статье, не являются стопроцентной панацеей, но все же они помогут вам самостоятельно перенести на Linux многие нужные Windows-приложения.