/etc/system.conf

Файл `/etc/system.conf` связывает сервис и список ресурсов, доступ к которым декларируется разработчиком сервиса или природой задачи. Необходимость конфигурации драйверов и других сервисов назрела уже давно. Подобного рода механизм вносит определённость в работу процессов, имеющих практически прямой доступ к ресурсам системы, которые возможно затребовать. Это ограничение нужно с точки зрения надёжности и безопасности.

Необходимость настроек с точки зрения надёжности заключается в следующем. Администратор, добавляя сервис, указывает ресурсы, с которыми должен контактировать сервис. Если в силу ошибки автора сервиса будет выполнена попытка обратиться к постороннему ресурсу (которые, естественно, в настройках отсутствует), то эта попытка сервиса потерпит неудачу.

Необходимость настроек с точки зрения безопасности заключается в следующем. Если какая-нибудь программа на компьютере сквозь брешь в системе получила достаточно прав для запуска себя в качестве сервиса, то одного этого будет недостаточно для выполнения хоть каких-либо действий. Должна быть ещё и запись в файле настроек сервисов, где её может обнаружить администратор.

Если система обнаружит несанкционированный доступ к ресурсу, то сообщит об этом в `/var/log/messages` и в первую консоль.

И последнее, о чём стоит упомянуть перед началом описания. Разбор содержимого файла выполняется средствами универсального транслятора конфигурционных файлов MINIX 3, поэтому, все неприятности, присутствующие при формировании других конфигурационных файлов, наблюдаются и здесь. На момент написания данного руководства одной из таких неприятных особенностей была ошибка разбора файла настроек при наличии в качестве отступов пробелов а не табуляций.

Описание построено на основе РБНФ: терминалы взяты в кавычки, нетерминалы начинаются с большой буквы, вертикальная черта обозначает выбор, фигурные скобки - повторение ноль и более раз, круглые скобки - неделимая конструкция. Если кто-либо найдёт более удобную основу - сообщайте на форум: Tutorial: Device Driver Programming in MINIX 3.

Структура файла `system.conf` состоит из множества секций сервисов:

```
ServiceConf = {Service ";"}.
```

Для каждого кодового файла, запускаемого с использованием утилиты service, необходима секция в данном файле. Если нужная секция не будет обнаружена, то будет выдана ошибка и, хотя сервис формально запустится (второй раз запустить не удаётся - имя этого драйвера уже зарегистрировано в системе), никаких действий выполнять он не может. Если секция будет пустой, то на экран никаких ошибок не выползет, но доступа к такому сервису всё равно не получить.

Секция сервиса имеет следующую структуру.

```
Service = "serivce" ServiceName "{" { Section ";" } "}".
```

`ServiceName` - имя сервиса. Оно не должно содержать пробелы и его длина должна укладываться в 16 символов. Остальное зависит от того, что считает лексемой универсальный транслятор конфигурационных файлов, которым пользуются для разбора `system.conf`. Если не указывать имя сервиса явно (при запуске `service`), то оно будет совпадать с именем кодового файла, указанного при запуске.

Секция доступа сервиса к ресурсам.

```
Section = ( Class | Uid | Nice | Irq | Io | Pci | System | Ipc | Vm
| Control ) ";".
```

Одна и та же секция доступа может встречаться несколько раз, но я бы не рекомендовал пользоваться данной особенностью: она скорее ухудшит читабельность, а не улучшит. К тому же, существуют секции, наличие которых в количестве более одного экземпляра сильно удивляет утилиту `service`.

Секция `Class` - это ссылка на секцию другого сервиса. А если быть точнее, то даже не сервиса, а контейнера для настроек, сгруппированных в одном месте. Причина, по которой контейнер начинается с ключевого слова `service`, не понятна.

```
Class = "class" ServiceName.
```

Указанное имя сервиса должно присутствовать в конфигурации.

В контейнере также может встретиться `Class`. Общее количество вложений не может превышать 100 (забито в коде).

Сервисы, имеющие много общих настроек, могут из своих секций разом указать эту группу настроек. А при необходимости добавить или удалить общие права у это группы сервисов достаточно будет сделать это в контейнере.

Коэффициент полезности данной секции крайне низок.

Секция `Uid` должна содержать имя пользователя или его номер в системе.

```
Uid = "uid" ( UsrLogin | UsrId ).
```

`UsrLogin` - имя пользователя в системе.

`UsrId` - идентификатор пользователя в системе.

Возможное назначение - наделить сервис правами этого пользователя.

Секция `Nice` содержит номер, задающий приоритет процесса.

```
Nice = "nice" Number.
```

`Number` - число.

Секция `Irq` содержит список irq, который, предположительно, должен обрабатываться сервисом.

```
Irq = "irq" { Number }.
```

Сквозной список[1]. Количество IRQ для одного сервиса должно быть <= 16.

Секция `Io` содержит список пар числовых значений, разделённых двоеточием.

```
Io = "io" { Number [ ":" Number ] }.
```

Предположительно, это отображённые на память порты для работы с оборудованием, к которому у сервиса должен быть доступ. Число до двоеточия - базовый адрес, а после двоеточия - размер области памяти. Оба числа - в 16-ной сиситеме счисления. Как видно из описания, длина может не указываться. В коде она никак не задаётся, поэтому потенциально может иметь любое значене (область переменной на стеке может содержат любой мусор(?)). Как я понимаю, данная секция должна использвоаться только драйверами.

Сквозной список. Количество адресов для одного сервиса должно быть <= 16.

1 Это значит, что при наличии в данной конфигурации нескольких секций этого вида, список ресурсов из новой секции не перезаписывает списко из старой, а добавляется в его хвост.

Секция Pci. Содержит описание устройства или класса.

```
Pci = "pci" [ ( PciDevice | PciClass ) ].
PciDevice = "device" { VendorId ["/" DeviceId] }.
PciClass = "class" { DevClass ["/" DevSubclass ["/" DevInterface]] }
```

PciDevice - PCI-устройство, описание которого представляет собой указание списка пар значений VendorId и DeviceId. Каждая пара описывает одно устройство. Сквозной список. Количество устройств для одного сервиса должно быть <= 32.

VendorId и DeviceId - числа в 16-ной системе счисления. Если DeviceId не задать, то он может оказаться произвольным.

PciClass - класс (устройств?), описание которого - это указание списка троек значений. Сквозной список. Количество классов устройств для одного сервиса должно быть <= 4.

DevClass, DevSubclass и DevInterface - числа в 16-ной системе счисления. По умолчанию DevSubclass и DevInterface равны нулю.

Секция Pci может быть пустой.

Секция System содержит список системных вызовов.

```
System = "system" { SysCallName }.

SysCallName = "EXIT" | "SYS_EXIT" | "PRIVCTL" | "TRACE" | "KILL" |
"UMAP" | "VIRCOPY" | "IRQCTL" | "INT86" | "DEVIO" | "SDEVIO" |
"VDEVIO" | "SETALARM" | "TIMES" | "GETINFO" | "SAFECOPYFROM" |
"SAFECOPYTO" | "VSAFECOPY" | "SETGRANT" | "READBIOS" | "PROFBUF" |
"STIME" | "MAPDMA" | "VMCTL" | "SYSCTL".
```

Список разрешённых SysCallName забит в код service.c и может смениться в любой момент и незаметно (если не мониторить service.c). Это список системных вызовов, которые разрешено выполнять сервису.

Секция Ipc содержит список имён процессов.

```
Ipc = "ipc" { ProcName }.
ProcName = Name.
```

Если данная секция встретится в конфигурации больше одного раза, то поизойдёт ошибка. С процессами из данного списка сервис сможет контактиовать (обмениваться сообщениями).

Секция Vm содержит список сообщений для виртуальной памяти.

```
Vm = "vm" { MsgName }.
MsgName = "REMAP" | "UNREMAP" | "GETPHYS" | "GETREFCNT" |
"QUERYEXIT" | "CTL".
```

Список названий MsgName забит в код service.c и может смениться в любой момент и незаметно (если не мониторить service.c).

Секция Control содержит список сервисов, которые данный сервис может перезапускать.

```
Control = "control" { ServiceName }
```

Количество для одного сервиса должно быть <= 8. Поскольку речь идёт об управлении сервисами другим сервисом, то естественно, что все они должны иметь свои настройки в system.conf. По крайней мере, в других случаях попытка перезапуска просто не сработает. Указываются имена сервисов как они есть в файле system.conf, а не кодовые файлы, предъявляемые утилите service при запуске сервиса.