README file for the osst driver
==========

(w) Kurt Garloff <garloff@suse.de> 12/2000

This file describes the osst driver as of version 0.8.x/0.9.x, the released version of the osst driver.
It is intended to help advanced users to understand the role of osst and to get them started using (and maybe debugging) it.
It won't address issues like "How do I compile a kernel?" or "How do I load a module?", as these are too basic.
Once the OnStream got merged into the official kernel, the distro makers will provide the OnStream support for those who are not familiar with hacking their kernels.

Purpose

The osst driver was developed, because the standard SCSI tape driver in Linux, st, does not support the OnStream SC-x0 SCSI tape. The st is not to blame for that, as the OnStream tape drives do not support the standard SCSI command set for Serial Access Storage Devices (SASDs), which basically corresponds to the QIC-157 spec.
Nevertheless, the OnStream tapes are nice pieces of hardware and therefore the osst driver has been written to make these tape devs supported by Linux.
The driver is free software. It's released under the GNU GPL and planned to be integrated into the mainstream kernel.

Implementation

The osst is a new high-level SCSI driver, just like st, sr, sd and sg. It can be compiled into the kernel or loaded as a module.
As it represents a new device, it got assigned a new device node: /dev/osstX are character devices with major no 206 and minor numbers like the /dev/stX devices. If those are not present, you may create them by calling Makedevs.sh as root (see below).
The driver started being a copy of st and as such, the osst devices' behavior looks very much the same as st to the userspace applications.

History

In the first place, osst shared its identity very much with st. That meant that it used the same kernel structures and the same device node as st.
So you could only have either of them being present in the kernel. This has been fixed by registering an own device, now.
st and osst can coexist, each only accessing the devices it can support by themselves.

Installation

osst got integrated into the linux kernel. Select it during kernel configuration as module or compile statically into the kernel.
Compile your kernel and install the modules.

Now, your osst driver is inside the kernel or available as a module, depending on your choice during kernel config. You may still need to create the device nodes by calling the Makedevs.sh script (see below) manually.

To load your module, you may use the command
modprobe osst
as root. dmesg should show you, whether your OnStream tapes have been recognized.

If you want to have the module autoloaded on access to /dev/osst, you may add something like
alias char-major-206 osst
to your /etc/modprobe.conf (before 2.6: modules.conf).

You may find it convenient to create a symbolic link
ln -s nosst0 /dev/tape
to make programs assuming a default name of /dev/tape more convenient to use.

The device nodes for osst have to be created. Use the Makedevs.sh script attached to this file.

Using it

You may use the OnStream tape driver with your standard backup software, which may be tar, cpio, amanda, arkeia, BRU, Lone Tar, ...
by specifying /dev/(n)osst0 as the tape device to use or using the above symlink trick. The IOCTLs to control tape operation are also mostly supported and you may try the mt (or mt_st) program to jump between filemarks, eject the tape, ...

There's one limitation: You need to use a block size of 32kB.

(This limitation is worked on and will be fixed in version 0.8.8 of
 this driver.)

If you just want to get started with standard software, here is an example for creating and restoring a full backup:
```
# Backup
tar cvf - / --exclude /proc | buffer -s 32k -m 24M -B -t -o /dev/nosst0
# Restore
buffer -s 32k -m 8M -B -t -i /dev/osst0 | tar xvf - -C /
```

The buffer command has been used to buffer the data before it goes to the tape (or the file system) in order to smooth out the data stream and prevent the tape from needing to stop and rewind. The OnStream does have an internal buffer and a variable speed which help this, but especially on writing, the buffering still proves useful in most cases. It also pads the data to guarantees the block size of 32k. (Otherwise you may pass the -b64 option to tar.)
Expect something like 1.8MB/s for the SC-x0 drives and 0.9MB/s for the DI-30. The USB drive will give you about 0.7MB/s.
On a fast machine, you may profit from software data compression (z flag for tar).

USB and IDE

Via the SCSI emulation layers usb-storage and ide-scsi, you can also use the osst driver to drive the USB-30 and the DI-30 drives. (Unfortunately, there is no such layer for the parallel port, otherwise the DP-30 would work as well.) For the USB support, you need the latest 2.4.0-test kernels and the latest usb-storage driver from
http://www.linux-usb.org/
http://sourceforge.net/cvs/?group_id=3581

Note that the ide-tape driver as of 1.16f uses a slightly outdated on-tape format and therefore is not completely interoperable with osst tapes.

The ADR-x0 line is fully SCSI-2 compliant and is supported by st, not osst. The on-tape format is supposed to be compatible with the one used by osst.

Feedback and updates

The driver development is coordinated through a mailing list
<osst@linux1.onstream.nl>
a CVS repository and some web pages.
The tester's pages which contain recent news and updated drivers to download can be found on
http://linux1.onstream.nl/test/

If you find any problems, please have a look at the tester's page in order to see whether the problem is already known and solved. Otherwise, please report it to the mailing list. Your feedback is welcome. (This holds also for reports of successful usage, of course.)
In case of trouble, please do always provide the following info:
* driver and kernel version used (see syslog)
* driver messages (syslog)
* SCSI config and OnStream Firmware (/proc/scsi/scsi)
* description of error. Is it reproducible?
* software and commands used

You may subscribe to the mailing list, BTW, it's a majordomo list.

Status

0.8.0 was the first widespread BETA release. Since then a lot of reports have been sent, but mostly reported success or only minor trouble.
All the issues have been addressed.
Check the web pages for more info about the current developments.
0.9.x is the tree for the 2.3/2.4 kernel.

Acknowledgments

The driver has been started by making a copy of Kai Makisara's st driver. Most of the development has been done by Willem Riede. The presence of the userspace program osg (onstreamsg) from Terry Hardie has been rather helpful. The same holds for Gadi Oxman's ide-tape support for the DI-30.

I did add some patches to those drivers as well and coordinated things a little bit.
Note that most of them did mostly spend their spare time for the creation of this driver.
The people from OnStream, especially Jack Bombeeck did support this project and always tried to answer HW or FW related questions. Furthermore, he pushed the FW developers to do the right things.
SuSE did support this project by allowing me to work on it during my working time for them and by integrating the driver into their distro.

More people did help by sending useful comments. Sorry to those who have been forgotten. Thanks to all the GNU/FSF and Linux developers who made this platform such an interesting, nice and stable platform.
Thanks go to those who tested the drivers and did send useful reports. Your help is needed!

Makedevs.sh

```
#!/bin/sh
# Script to create OnStream SC-x0 device nodes (major 206)
# Usage: Makedevs.sh [nos [path to dev]]
# $Id: README.osst.kernel,v 1.4 2000/12/20 14:13:15 garloff Exp $
major=206
nrs=4
dir=/dev
test -z "$1" || nrs=$1
test -z "$2" || dir=$2
declare -i nr
nr=0
test -d $dir || mkdir -p $dir
while test $nr -lt $nrs; do
  mknod $dir/osst$nr c $major $nr
  chown 0.disk $dir/osst$nr; chmod 660 $dir/osst$nr;
  mknod $dir/nosst$nr c $major $[nr+128]
  chown 0.disk $dir/nosst$nr; chmod 660 $dir/nosst$nr;
  mknod $dir/osst${nr}l c $major $[nr+32]
  chown 0.disk $dir/osst${nr}l; chmod 660 $dir/osst${nr}l;
  mknod $dir/nosst${nr}l c $major $[nr+160]
  chown 0.disk $dir/nosst${nr}l; chmod 660 $dir/nosst${nr}l;
  mknod $dir/osst${nr}m c $major $[nr+64]
  chown 0.disk $dir/osst${nr}m; chmod 660 $dir/osst${nr}m;
  mknod $dir/nosst${nr}m c $major $[nr+192]
  chown 0.disk $dir/nosst${nr}m; chmod 660 $dir/nosst${nr}m;
  mknod $dir/osst${nr}a c $major $[nr+96]
  chown 0.disk $dir/osst${nr}a; chmod 660 $dir/osst${nr}a;
  mknod $dir/nosst${nr}a c $major $[nr+224]
  chown 0.disk $dir/nosst${nr}a; chmod 660 $dir/nosst${nr}a;
  let nr+=1
done
```