OpenServer Kernel Personality (OKP)

White Paper

March, 2004

Overview of OpenServer Kernel Personality (OKP)

OpenServer Kernel Personality for SCO UnixWare enables installation and native execution of OpenServer applications on SCO UnixWare. OKP combines the UnixWare Application Compatibility Package with a full OpenServer environment to provide the highest level of compatibility with OpenServer applications.

A UnixWare system with OKP can run a wide range of OpenServer programs. Programs like Communicator and Mozilla and tools that are unique to OpenServer - including SCOterm, SCOsh, and SCOpaint - function perfectly. Applications available only for Xenix, such as Megabasic and Progress 7 also run using OKP.

OKP harnesses the power and scalability of a single UnixWare system to run OpenServer and UnixWare applications side-by-side. The benefits of this approach are:

- The OKP feature can be used to reduce operating expenses by implementing a low-risk server consolidation project.
- Running OpenServer applications on OKP gives you the option of the failover clustering available with ReliantHA® on UnixWare.
- OKP may increase the performance of existing OpenServer applications because up to 32 processors per system are supported with OKP on UnixWare.
- With OKP on UnixWare you can continue to use you existing applications while taking advantage of UnixWare's support for new hardware.

Application Compatibility Package

The base UnixWare operating system already runs OpenServer binaries through its Enhanced Application Compatibility package (ACP). In addition, UnixWare includes a selection of OpenServer tools and utility programs in its /OpenServer/bin directory, along with supporting libraries in /OpenServer/lib and /OpenServer/usr/lib.

At first glance, it appears that UnixWare is fully capable of running any OpenServer application without any further "tweaking". However, ACP does have limitations:

- Limited OpenServer environment
- Difficult to upgrade
- Lack of migration tools

OKP addresses these limitations by providing a more complete OpenServer environment.

Limited OpenServer environment

Whenever an OpenServer program runs, UnixWare must make sure that it links the correct shared libraries. ACP handles this by pre-pending the shared library search path with */OpenServer* when an OpenServer program is detected.

This is a reasonable approach for simple cases, but starts to break down where libraries exist in UnixWare, but not in OpenServer. In some cases ACP will find the wrong library because there is no strong separation of the UnixWare and OpenServer environment.

Both UnixWare and OpenServer binaries specify /usr/lib/libc.so.1 as their ELF interpreter, the kernel can't readily tell an OpenServer binary from a UnixWare one. So it hands both to the main system libc. This library has the job of trying to detect an OpenServer program, and on seeing one, shunts control to the OpenServer C library located in the /OpenServer directory.

Difficult to upgrade

To make things even more complicated, this approach needs UnixWare's libc and the OpenServer libc in /OpenServer/usr/lib to cooperate. This brings us to another limitation of ACP, ease of upgrading.

ACP's use of the /OpenServer directory to store special versions of libraries also creates an update headache. A patch or upgrade for OpenServer can be applied to OpenServer's C libraries when installed on an OpenServer system. This same patch cannot, however, be applied to UnixWare's /OpenServer directory, often meaning that UnixWare's /OpenServer/usr/lib/libc.so.1 is doomed to be almost perpetually out of date. This breaks support for any applications that require the latest of OpenServer's C library. To upgrade an administrator must manually copy updated C libraries to the /OpenServer directory, then create any necessary links. This method is tedious, and error prone.

Lack of migration tools

ACP doesn't have tools to ease the transition of applications from OpenServer to UnixWare. All configuration and installation tasks associated with installing an application will have to be done manually. This might also include making many links between libraries in the /OpenServer directory to where the applications libraries are located. It is not as simple as just directly copying files from an OpenServer server to the new UnixWare server.

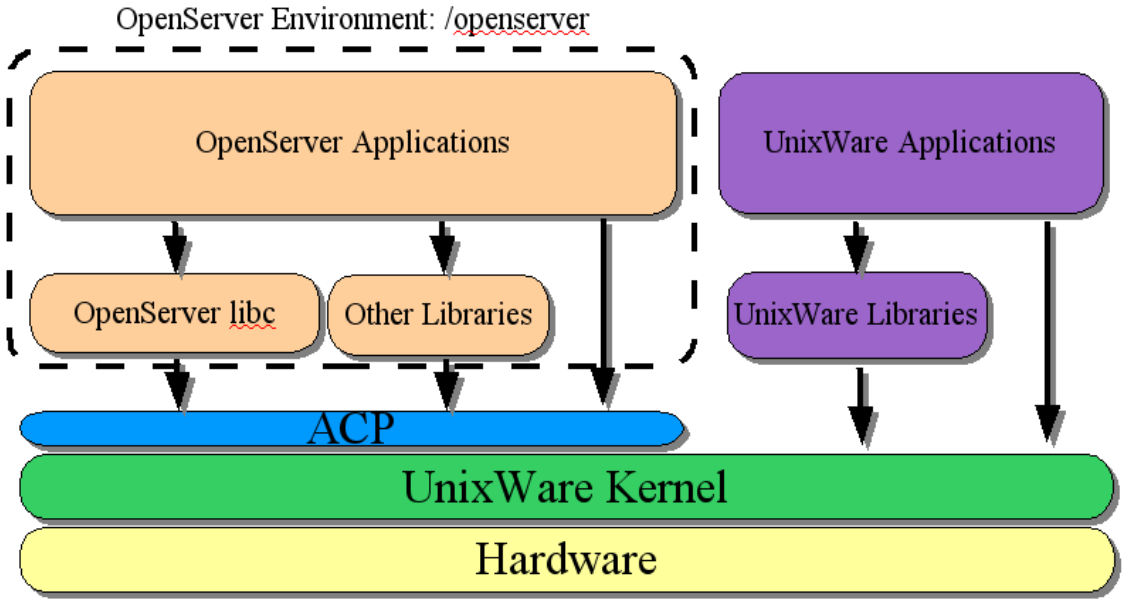
Benefits of the OKP Approach

OKP addresses the pitfalls and limitations of ACP through the following:

- OKP reworks user level ACP
- OKP extends the kernel level ACP
- Uname is modified.

OKP reworks user level ACP

The user level ACP uses the same environment as UnixWare, so to solve any library conflicts, OKP creates a clean user-space environment using the chroot utility. This same method is used by the Linux Kernel Personality (LKP) to separate its operating environment from UnixWare's. There is a directory, /openserver, created on an OKP system which is similar to the /linux directory



used for LKP. The /openserver directory contains a complete copy of a whole OpenServer system root directory. In this way, OKP sidesteps the entire user-space C library shunting that happens in ACP. Like any other OpenServer binary, these tools are not aware that they are not running on native OpenServer. This lets the user take advantage of many features of native OpenServer. For example, OpenServer init scripts run without modification.

OpenServer style internationalization and localization are mapped to the corresponding UnixWare 7 locales by the openserver command used to execute OpenServer applications under OKP. This creates a complete OpenServer environment, and increases application compatibility.

The directory is named /openserver for consistency with LKP, but this can be confusing. It is very easy to mix up /openserver and /OpenServer. However, it is necessary to have both directories as the first is OKP's compatibility directory; the second is ACP's.

OKP extends kernel level ACP

This approach by OKP also retains all of the kernel level pieces of ACP. In fact, it absolutely relies on them. An OpenServer program running under OKP must see the same special behaviors from the kernel that programs see when running under ACP. Even though an application is run from the OpenServer environment, the executable is still handed to the UnixWare kernel for execution. It is up to the kernel to determine if the application is an OpenServer or UnixWare executable. The kernel does this automatically after it reads the executable's header information. From the header information the kernel knows if the executable is a COFF binary, or an OpenServer ELF binary. Once the kernel has determined the executable is an OpenServer executable, the ACP kernel routines are used to execute the binary.

Modifying Uname

Uname is a system call that a process can use to find out details about the system it is running on. OpenServer processes should not even be aware that they are not running on native OpenServer. UnixWare's uname returns OpenServer-like results for OpenServer processes that call it.

So now we have a system which allows us to change root into /openserver, and from in here, run OpenServer binaries seamlessly. We use the kernel compatibility part of UnixWare's ACP, but the user-space parts are ignored, staying well away from the /OpenServer/usr/lib ACP OpenServer libraries.

Software Development Model

OKP makes it possible to develop and deploy applications on systems larger than the current scalability limits for OpenServer. Applications can take advantage of UnixWare's excellent device support, support for up to 32 processors per system, increased file sizes, up to 16 GB of memory, and a multi threaded journaling file system. It also allows an application to take advantage of several high availability applications, such as ReliantHA®, to provide a higher level of application, system and data availability. All this can be accomplished without having to port applications to a new platform. You can move forward and plan for the future of your infrastructure while still being able to run existing OpenServer or legacy Xenix application suites.

Application Compatibility

A large number of OpenServer applications will be able to run on OKP without configuration changes. There are some cases where an OpenServer application will not run correctly on OKP. The most common reason is that OpenServer and UnixWare handle hardware devices differently. OKP relies on UnixWare's hardware drivers. For example the `custom` utility, when invoked in the OKP environment, cannot directly access a CDROM drive. The work around this problem is to mount the CDROM device in a UnixWare shell before invoking custom from the OpenServer environment. To mount the CDROM device so it is visible to OKP, you can use the following command:

```
/sbin/mount /dev/cdrom/cdrom1 /openserver/mnt
```

Another case where some configuration changes may need to be made is when running terminal based OpenServer applications. Several text based applications do not display correctly on OKP due to differences between the UnixWare and OpenServer terminal settings. Key-mappings might not correct for these applications as well. There are several ways to fix this problem, should it come up. The first is to change the default console font using the following command:

```
mv /etc/default/cofont /etc/default/confont.old
```

Several other terminal settings may also need to be changed to get the application to display correctly. Configuring the terminal settings is beyond the scope of this document. For more information refer to the SCO support page: http://www.thescogroup.com/support.

To solve the keyboard mapping problems use the scostrings keyboard mappings as the default keyboard map:

```
mv /usr/lib/keyboard/strings /usr/lib/keyboard/strings.old
cp /usr/lib/keyboard/scostrings /usr/lib/keyboard/strings
```

The above commands will change how the keyboard works with regular UnixWare applications. This is a current limitation that we plan to address in future versions of OKP.

Another known limitation is that any Administration utility written to use the OpenServer scoadmin interface will have to be re-written to UnixWare's scoadmin interface. This includes the graphical and non-graphical Administration interface.

In addition to the scoadmin utilities several common OpenServer utilities check for a valid OpenServer license before they will run. These applications include:

make, the calendar daemon, and the OpenServer desktop Xdt3.
These programs will not currently work under OKP on UnixWare because the UnixWare licensing daemon, pmd, is operating on the system. Because the UnixWare pmd is holding the licensing socket, the OpenServer pmd is unable to run under OKP. The UnixWare pmd is unable to process requests from OpenServer applications. OpenServer's pmd is also unable to run under OKP because it needs some low-level information off the disk that UnixWare doesn't provide.

There are other cases where an OpenServer application will not work under OKP. The quickest way to see what is causing the application to not start, or work correctly is to use the `osrtruss` utility. To run an application under `osrtruss` from a UnixWare shell:

```
cd /openserver/<app_path>
osrtruss -f [truss_options] openserver "./myapp arg1 arg2 ..."
```

For more information on the `osrtruss` and `truss` command refer to the `truss(1)` manual page. For the latest fixes and solutions to problems refer to the SCO Support page: http://www.thescogroup.com

Device Compatibility

There has been some specific discussion on issues with certain applications and devices, however more generally OKP for UnixWare 7 supports all hardware currently supported by UnixWare 7 using existing drivers. Usually, accessing the device is transparent to the OpenServer application, however some applications use direct connections to devices. An example of this is an application accessing third-party multi-port serial devices. If there is a driver for the device for UnixWare then OKP should be able to access the device using the standard /dev file interface. Sometimes a hard link is necessary from the device file in the /dev directory to the device file the /openserver/dev directory. This allows the application to access the device file when running under chroot. Most of the time creating hardlinks between device files is not necessary as OpenServer I/O activities are mapped from the OpenServer system call via the device/kernel interface to UnixWare 7 I/O events.

As mentioned before OKP relies upon UnixWare hardware support for its hardware support. Hardware drivers written for OpenServer will not work using OKP. If special hardware is required to run an OpenServer application, before migrating to OKP, check to see if a hardware driver is available for UnixWare.

Operation

There are two general ways an OpenServer program is run on an OKP system.

The first is to start an OpenServer terminal, which creates an environment which is very similar to a native OpenServer environment. The second is to pass the application and arguments to the `openserver` command. Examples of these two methods are shown below.

Note: Now we will have to differentiate between the UnixWare terminal and the OKP terminal. For the UnixWare terminal the "$" symbol will be used. For the OpenServer terminal "[openserver]" will be used.

```
$ openserver
[openserver] cd application_path
[openserver] ./command arg1 arg2 ...
```

where **application_path** is the path to the application's startup directory, and **command** starts the applications. When you use the **openserver** command in this manner the root directory is changed to the /openserver directory, and the /openserver/etc/profile file is read to configure the environment. In the example above if the **command** is a directory specified by the $PATH environmental variable, then after entering the [openserver] terminal you would just have to enter the command.

For the convenience of launching applications from scripts the method to launch an OpenServer application is slightly different. For example to launch **command** from a script the following commands would be entered:

```
cd /openserver/application_path
openserver ./command arg1 arg2 ...
```

If the application is not part of the OpenServer image used to populate the /openserver directory, you can use OpenServer tools to install the application on OKP. Use 'custom' to install application packaged with 'custom' or CDMT. The OpenServer versions of tar and cpio are available from within OKP. Finally, for installing pkg* utilities use the native UnixWare pkgadd command.

For additional details on running applications using OKP refer to the OKP documentation which is part of the SCOhelp documentation system.

XENIX EMULATION

UnixWare cannot directly run Xenix binaries. However, some OpenServer applications really turn out to be Xenix applications, or sometimes contain a mixture of OpenServer and Xenix binaries. OKP uses the following process to cope with this:

Lcall7 call gate overload

OKP carries around a UnixWare kernel module, xout, whose job is to interface with the user space Xenix emulator. The kernel module is very small, and has two short, but very important, jobs to do:

It checks the magic number of the binary (the first two bytes) for the value 0x0206. If it finds this value, the binary is a Xenix one, so the user space Xenix emulator is invoked to run the binary. It opens an additional call gate for system calls at 0x37, in effect lcall37. This operates in parallel with lcall7. Lcall37 is an alias for lcall7; that is, both call gates transfer control to exactly the same location inside the kernel, in this case, the main system call handler function.

User space emulator

OKP comes with a user space Xenix emulator, /usr/bin/xrun, linked to /usr/bin/xemul. This is the program that the xout kernel module runs.

The emulator's first job is to redirect its own lcall7 call gate so that it points to a function inside its own code, not to the UnixWare kernel's system call handler. In this way it becomes its own system call hander.

Of course, this is not going to work unless the emulator has access to real system calls, and this is where the extra call gate, lcall37, comes in. The emulator redirects real system call requests to lcall37, and catches system call requests from the Xenix binary to lcall7. It becomes a system call "filter", catching, modifying if necessary, and passing on, system calls from Xenix binaries. It catches system calls to lcall7, and passes them on to lcall37.

The Xenix emulator is also responsible for memory mapping and segmentation on behalf of the Xenix program. It works very much like the LKP kernel module, except for the very real difference that it is not actually in the kernel.

For more information on xemul refer to the xemul(1) manual page.

Bypassing selected file permissions

If a user has execute permission for a binary file or a script, they should be able to run it, even if they have no read permission. However, the Xenix emulator needs to be able to read in a Xenix binary before it can run it. To do this successfully, it may need to bypass read permission for the file, in effect treating execute permission as read permission.

Running full OpenServer application suites

Once a Xenix program is loaded into the emulator, it runs as a first-class

system process. The UnixWare kernel does not differentiate between Xenix and UnixWare processes. In fact, the kernel really cannot tell the difference because the emulator is UnixWare code, and it is that code which is making all of the system calls.

Limitations of OKP

Some limitations of OKP have already been discussed, however when considering a migration to OKP some other items have to be considered.

Mail services

OpenServer mail services will not work under OKP, except for mail delivered to a local mailbox (i.e., on the same system). Mail services should be configured and administered under UnixWare using UnixWare utilities. Use one of the UnixWare mailers (pine, mailx, etc.) to send and receive mail.

File Systems

UnixWare 7 file system types are used for OKP. In the migration process, user data files are copied onto native UnixWare 7 file systems. Note that no direct access to OpenServer HTFS and DTFS filesystem architectures is provided under OKP. If an OKP application calls a filesystem administration command such as mount, mkfs, or fsck, the native UnixWare 7 version of the command is called rather than the OpenServer version.

Graphics

- scoterm is unable to run in scancode mode.
- Most application pixmaps, bitmaps, and masks from SCO OpenServer will not work.
- Motif User Interface Language (UIL) files are not portable from SCO OpenServer.
- Applications using IXI Drag and Drop will not work.
- The scoforeground/scobackround X resources from SCO OpenServer are not available.
- Some keyboard definitions provided by SCO OpenServer applications may not work.

Mass storage drivers

Some devices that were supported in SCO OpenServer may not be supported in UnixWare 7. These may include low-end non-SCSI devices such as floppy-tape. Support for such devices may be available from third parties, but this is not guaranteed. Devices that conform to SCSI or IDE/Atapi, with the exception of IDE Tape, should work in UnixWare 7. In addition:

- Device naming conventions are different in UnixWare 7 and compatibility of device node names is not guaranteed.
- Device drivers are not compatible.
- Mass storage ioctls are different in UnixWare 7.
- A different set of mass storage commands is provided.

Conclusion

OKP allows a business to leverage the power, scalability and availability of UnixWare while continuing to support existing OpenServer and Xenix applications. This method is very cost effective as the business does not need to incur the added cost of porting applications and data to UnixWare. Also, without the wait of porting applications and data, this solution is ready to go now, and can be implemented without a large amount of downtime. For more detailed information on running OKP refer to the OKP documentation, which is part of the SCOhelp system.