Tivoli software

IBM

Redbooks Paper

Deon George

Backing Up Databases using ADSMPIPE and the TSM API: Examples Using Linux

Introduction

This IBM® Redpaper describes how you can use the IBM Tivoli® Storage Manager (TSM) application programming interface (API) with the ADSMPIPE utility to perform online backups of various databases. This process eliminates dumping data to a file (and consuming disk space) so that the TSM backup and archive client can back up the data.

This paper uses examples of specific databases running on Linux® (MySQL, PostgreSQL, and OpenLDAP); however, the techniques apply equally for these and other databases running on any UNIX® operating system. They are intended for database applications that have no dedicated TSM applications available.

This procedure may be used for any database where you can dump the data to STDOUT and pipe it to TSM for storage. The TSM API stores the STDOUT data in your TSM storage hierarchy and optionally compresses the data before sending it to TSM.

The configuration files used to implement the features described in this paper can be accessed as follows

- You may download them from:

  ftp://www.redbooks.ibm.com/redbooks/REDP3980/

- You may also go to this Web site:

  http://ibm.com/redbooks

  From there:

  a. Select **Additional materials**.

  b. Open the directory that corresponds with the IBM Redpaper form number, REDP3980.

There are two downloadable files:

- adsmpipe.tar.Z (the source files for the adsmpipe utility)
- 3980scripts.zip (other script files and configuration files referred to in this Redpaper)

Important: The adsmpipe utility and the scripts are provided “as-is.” In other words, they are not supported by IBM. The scripts must be customized to work in your particular environment.

Preparation and requirements

To use the TSM API, you need the following:

- Access to the Linux systems database that you want to back up (for example, MySQL, PostgreSQL, or OpenLDAP)
- Depending on your configuration, a user ID and password to get access to the database with the privileges required to read all the data for backup and write to the database when performing a restore
- Access to your Linux systems file system to install the adsmpipe utility

  Note: We recommend having root authority for this, so that you can install it in the directory /usr/bin for all users.

- Client access to a TSM server, either using your backup and archive client login and a password or by registering a special node for the adsmpipe data
- TSM API (the version used in this paper was V5.2.3), which can be downloaded from:

  `http://www-1.ibm.com/support/docview.wss?rs=663&context=SSGSG7&dc=D42G3&uid=swg24008552&loc=en_US&cs=utf-8&lang=en`
- A system with gcc and make installed, so that you can compile the adsmpipe utility. You should compile the adsmpipe utility for each Linux system, where:
  - The version of glibc is different.
  - The version of the TSM API is different

There is a sample .spec file and patch file in “Building an RPM for adsmpipe” on page 13. You can use it to build Red Hat Package Manager (RPM), which may help you deploy adsmpipe to your TSM clients.

Downloading and building adsmpipe

Download the **adsmpipe** source files from this Web site:

`http://www.redbooks.ibm.com/redbooks/REDP3980/adsmpipe.tar.Z`

The **adsmpipe** utility is a special *unsupported* TSM client that takes input from STDIN and can send that data to TSM for storage as either backup or archive data.

Data stored in TSM as *backup* data is retained according to the management class policy, retaining extra versions where the policy can define the number of versions or number of days (or both) to keep the extra copies.

Data stored in TSM as *archive* data is also retained according to the management class policy, where the policy can determine the number of days to keep the archived data.

Follow these steps to build **adsmpipe**:

1. Extract the **adsmpipe** source code from the compressed tar archive as shown in Example 1.

Example 1 Extract adsmpipe files

```
deon@dgeorge-lnx tmp]$ tar xvzf adsmpipe.tar.Z
aix/
aix/ver1/
aix/ver1/aixapi.v1r2m7.README
aix/ver1/aixapi.v1r2m7.smit
aix/ver2/
aix/ver2/libApiDS.a
aix/ver2/dscameng.txt
aix/ver2/dsmapitca
aix/adsmpipe/
aix/adsmpipe/Makefile
aix/adsmpipe/README
aix/adsmpipe/adsmblib.c
aix/adsmpipe/adsmpipe.c
aix/db2bkdel/
aix/db2bkdel/db2bkdel
solaris/
solaris/ver1/
solaris/ver1/sunapi.v1r2m7.README
solaris/ver1/sunapi.v1r2m7.tar.Z
solaris/ver2/
solaris/ver2/dscameng.txt
solaris/ver2/dsmapitca
solaris/ver2/libApiDS.so.1.sol2
solaris/adsmpipe/
solaris/adsmpipe/Makefile
solaris/adsmpipe/README
solaris/adsmpipe/adsmblib.c
solaris/adsmpipe/adsmpipe.c
[deon@dgeorge-lnx tmp]$
```

2. Change to the aix/adsmpipe directory. Since there is no explicit Linux version of **adsmpipe**, you can use the IBM AIX® version for compiling as follows:

```
[deon@dgeorge-lnx tmp]$ cd aix/adsmpipe
[deon@dgeorge-lnx adsmpipe]$
```

3. Edit the Makefile and change the CFLAGS line to include the path to TSM API client header files (this is where the libApiDS.a, dsmapifp.h, dsmapitd.h, dsmrc.h files are located). For example, change: `CFLAGS=-g to:`

```
CFLAGS=-g -I/opt/tivoli/tsm/client/api/bin/sample
```

4. Run **make** to compile the **adsmpipe** utility. It should look similar to the output shown in Example 2.

Example 2 Compile adsmpipe

```
[deon@dgeorge-lnx adsmpipe]$ make
cc -g -I/opt/tivoli/tsm/client/api/bin/sample   -c -o adsmpipe.o adsmpipe.c
cc -g -I/opt/tivoli/tsm/client/api/bin/sample   -c -o adsmblib.o adsmblib.c
cc -o adsmpipe adsmpipe.o adsmblib.o -lApiDS -L.
rm *.o
[deon@dgeorge-lnx adsmpipe]
```

5. You now have an **adsmpipe** binary. You can verify this as shown in Example 3.

Example 3 Check that the adsmpipe binary exists

```
[deon@dgeorge-lnx adsmpipe]$ ls -al
total 253
drwxr-x---  2 deon admin   216 Sep 15 16:19 .
drwxr-x---  6 deon admin   144 Dec 19  1995 ..
-rw-r-----  1 deon admin 31446 Dec 16  1995 adsmblib.c
-rwxr-xr-x  1 deon admin 86685 Sep 15 16:19 adsmpipe
   -rw-r-----  1 deon admin  7185 Dec 16  1995 adsmpipe.c
   -rw-r-----  1 deon admin   312 Sep 15 16:19 Makefile
   -rw-r-----  1 deon admin 30000 Dec 19  1995 README
   [deon@dgeorge-lnx adsmpipe]$
```

6. Copy the **adsmpipe** executable to a suitable directory (for example, /usr/bin or /usr/local/bin) in the normal user path.
7. To show the detailed syntax for using **adsmpipe**, enter **adsmpipe** on the command line without any options as shown in Example 4.

Example 4 Syntax for adsmpipe

```
adsmpipe [-[tcxd] -f <filename> [-l<size>] [-v]] [-p <oldpw/newpw/newpw>]

Creates, extracts or lists files in the ADSM pipe backup area
        -t      Lists files in pipe backup area matching the pattern
        -c      Creates a file in pipe backup area.
                 Data comes from standard input.
        -x      Extracts a file in pipe backup.
                 Data goes to standard output.
        -d      Deletes the file from active store.

        -B      Store data in the backup space of the ADSM server
        -A      Store data in the archive space of the ADSM server
        -f <file>
                Provides filename for create, and extract operations.
                 For list operations, the filename can be a pattern
        -l <size>
                Estimated size of data in bytes.
                 This is only needed for create
        -p <oldpw/newpw/newpw>
                <oldpw/newpw/newpw> Changes password
                 <passwd> Uses passwd for signon
        -v      Verbose
        -m      Overwrite management class (API Version 2.1.2 and higher)
        -n <count>
                File number to retrieve if multiple versions
        -s <filespace>
                Specify file space (default"/adsmpipe")
```

Backing up MySQL

MySQL provides a utility called **mysqldump** that dumps data from a running MySQL database, in ASCII format, to STDOUT. Refer to the man page for mysqldump to obtain the full syntax for this command, as well as other information.

You can back up a MySQL database by running **mysqldump** and piping the output to **adsmpipe**, with a statement such as:

```
mysqldump [[-u <username>] [-p <password>]] [-h <mysqlhost>] [<database>] | adsmpipe -f
/mysqldb -c
```

where:

`-u <username>` is a user ID that has full SELECT privileges (if required).

`-p <password>` is a password for the user ID (if required).

`<database>` is a name of a database.

You may use `--all-databases` for all databases.

Example 5 shows a more advanced script that you can use to back up MySQL. This script:

- Backs up all databases, each with their own TSM file name
- Stores all MySQL databases in the TSM /adsmpipe/mysql file space
- Expires any databases stored in TSM that are no longer in MySQL

Example 5 Sample of a full script for backing up MySQL databases

```
#!/bin/sh

#$Header: /tdpLinux/mysql.backup.sh,v 1.4 2004/09/21 01:08:47 deon Exp $

# This script will perform an MYSQLDUMP to TSM.

if [ -r /etc/sysconfig/tdpLinux.conf ]; then
        . /etc/sysconfig/tdpLinux.conf

else
        #VERIFY="n"
        VERIFY_TMP="/tmp/"
        #VERBOSE="-v"
        LOGFILE="/tmp/tdpLinux.log"

        # MYSQL Settings, make sure your user that you define here has access to all
databases
        # passed via the first command line argument.
        MYSQL_SERVER="localhost"
        MYSQL_USER="root"
        #MYSQL_PASSWD="password"

        # Set our path so that all our commands are available.
        PATH=$PATH:/usr/local/bin:/usr/bin:.
fi

# The high level file system identifier.
ADSMPIPE_PATH="/adsmpipe/mysql"

DB="$1"; shift
RESULT=0
SCRIPT=$(basename $0)

# Skip the first line of mysqldump, it contains the word "Database".
skipFirst=1;

for i in $(echo show databases | mysql ${MYSQL_SERVER:+-h $MYSQL_SERVER} ${MYSQL_USER:+-u
$MYSQL_USER} ${MYSQL_PASSWD:+--password=$MYSQL_PASSWD}); do

        # Skip the first line from the output, it is an SQL header.
        [ $skipFirst -eq 1 ] && skipFirst=0 && continue;
```

```
        echo $i >> /tmp/$SCRIPT.$$

        # If we supplied a database on the command line, than just back that one up.
        [ -n "$DB" -a "$DB" != "$i" ] && continue;

        [ -n "$VERBOSE" ] && echo "BACKING up database [$i] $(date)" >> $LOGFILE
        DB_CMD="mysqldump -B -F --add-drop-table ${MYSQL_SERVER:+-h $MYSQL_SERVER}
${MYSQL_USER:+-u $MYSQL_USER} ${MYSQL_PASSWD:+--password=$MYSQL_PASSWD} $i"
        if [ -n "$VERIFY" ]; then
                $DB_CMD 2>> $LOGFILE | tee /${VERIFY_TMP}/$i.tsm | adsmpipe -f ${i} -c -s
${ADSMPIPE_PATH} ${VERBOSE:-} >> $LOGFILE 2>&1
        else
                $DB_CMD 2>> $LOGFILE | adsmpipe -f ${i} -c -s ${ADSMPIPE_PATH} ${VERBOSE:-}
>> $LOGFILE 2>&1
        fi

        RETURNCODE=$?

        if [ -n "$VERIFY" -a $RETURNCODE -eq 0 ]; then
                echo "- Verify DB checksum [$(md5sum /${VERIFY_TMP}/$i.tsm|awk '{print
$1}')]" >> $LOGFILE 2>&1
                echo "- Verify TSM checksump [$(adsmpipe -f ${i} -x -s ${ADSMPIPE_PATH} |
md5sum|awk '{print $1}')]" >> $LOGFILE 2>&1
                rm -f /${VERIFY_TMP}/$i.tsm
        fi

        [ -n "$VERBOSE" ] && echo "= Backing COMPLETED with code [$RETURNCODE] $(date)" >>
$LOGFILE

        # If there was an error, then report it
        [ $RETURNCODE -gt 0 ] && echo "! ERROR [${RETURNCODE}] while backing up database
[${i}]" >> $LOGFILE && RESULT=1

done

# Now list all our backups, and see which ones we need to expire.
for i in $(adsmpipe -f \* -t -s ${ADSMPIPE_PATH} 2>&1|awk '{if ($2 == "(A)") {print $5}}');
do

        # Chop off the first slash
        i=${i#/}

        # If we supplied a database on the command line, than just work on that one.
        [ -n "$DB" -a "$DB" != "$i" ] && continue;

        [ -n "$VERBOSE" ] && echo "- Checking DB [$i]" >> $LOGFILE

        if grep -q $i /tmp/$SCRIPT.$$ ; then
                [ -n "$VERBOSE" ] && echo "- DB [$i] is still current." >> $LOGFILE
        else
                [ -n "$VERBOSE" ] && echo "- DB [$i] is no longer current - expiring..." >>
$LOGFILE
                adsmpipe -f $i -d -s ${ADSMPIPE_PATH} ${VERBOSE:-} >> $LOGFILE 2>&1
        fi

done

rm -f /tmp/$SCRIPT.$$
```

```
exit $RESULT
```

Restoring MySQL

You must consider how the data was backed up when you are restoring MySQL. The **mysqldump** command has options for:

- Backing up all databases in one dump or in an individual database
- Including `create database` in the SQL statements in the dump
- Including `add drop table` in the SQL statements in the dump

Once you have determined how the data was sent to TSM, the restore process is the reverse of the backup; that is, you extract the backup with **adsmpipe**, then direct the restored dump output to the **mysqldump** command as follows:

```
adsmpipe -f /mysqldb -x | mysql [[-u <username>] [-p <password>]] [-h <mysqlhost>]
[<database>]
```

If you used the script shown in Example 5 on page 5, then you may use the corresponding restore script as shown in Example 6.

Example 6 Sample of a full script for restoring MySQL databases

```
#!/bin/sh

#$Header: /tdpLinux/mysql.restore.sh,v 1.2 2004/09/20 06:02:15 deon Exp $

# This script will restore a MYSQL database from TSM.

if [ -r /etc/sysconfig/tdpLinux.conf ]; then
        . /etc/sysconfig/tdpLinux.conf

else
        #VERIFY="n"
        VERIFY_TMP="/tmp/"
        #VERBOSE="-v"
        LOGFILE="/tmp/tdpLinux.log"

        # MYSQL Settings, make sure your user that you define here has access to all
databases
        # passed via the first command line argument.
        MYSQL_SERVER="localhost"
        MYSQL_USER="root"
        #MYSQL_PASSWD="password"

        # Set our path so that all our commands are available.
        PATH=$PATH:/usr/local/bin:/usr/bin:.
fi

# The high level file system identifier.
ADSMPIPE_PATH="/adsmpipe/mysql"

DB="$1"; shift
RESULT=0

for i in $(adsmpipe -t -f /\* -s $ADSMPIPE_PATH 2>&1 | awk '{if ($2 == "(A)") {print
$5}}'); do

        i=${i#/}
```

```
        # If we supplied a database on the command line, than just restore that one.
        [ -n "$DB" -a "$DB" != "$i" ] && continue;

        [ -n "$VERBOSE" ] && echo "RESTORING database [$i] $(date)" >> $LOGFILE

        DB_CMD="mysql ${MYSQL_SERVER:+-h $MYSQL_SERVER} ${MYSQL_USER:+-u $MYSQL_USER}
${MYSQL_PASSWD:+--password=$MYSQL_PASSWD} mysql"

        adsmpipe -f ${i} -x -s ${ADSMPIPE_PATH} ${VERBOSE:-} 2>> $LOGFILE | $DB_CMD 2>>
$LOGFILE 2>&1

        RETURNCODE=$?

        [ -n "$VERBOSE" ] && echo "= Restore COMPLETED with code [$RETURNCODE] $(date)" >>
$LOGFILE

        # If there was an error, then report it
        [ $RETURNCODE -gt 0 ] && echo "! ERROR [${RETURNCODE}] while restoring database
[${i}]" >> $LOGFILE && RESULT=1
done

exit $RESULT
```

Backing up PostgreSQL

PostgreQSQL provides two utilities, **pg_dump** and **pg_dumpall**, to dump data from a running PostgreSQL database to STDOUT. Using **pg_dump** dumps out data one database at a time; **pg_dumpall** dumps all databases.

Use **pg_dump** or **man pg_dump** to see the full syntax for these utilities.

You can back up PostgreSQL databases by running the **pg_dumpall** command and piping the output to **adsmpipe** with a statement such as:

```
pg_dumpall -C [-U <username>] [-h <pgsqlhost>] | adsmpipe -f /pgsqldb -c
```

where:

-C adds create table syntax to the dump.

-U <username> is a user ID that has full SELECT privileges (if required).

Note: Unfortunately, **pg_dumpall** will prompt the user for password if the database server has been configured to authenticate users. Keep that in mind if you want to use **pg_dumpall** in a script.

The full syntax for **adsmpipe** was shown in Example 4 on page 4. You can back up PostgreSQL databases with a more advanced script that:

- Backs up all databases, each with their own TSM file name
- Stores all PostgreSQL databases in the /adsmpipe/pgsql file space in TSM
- Expires any databases stored in the TSM that are no longer in PostgreSQL

This script is shown in Example 7.

Example 7 Sample of the full script for backing up PostgreSQL databases

```
#!/bin/sh
```

```
#$Header: /san/CVS/tdpLinux/pgsql.backup.sh,v 1.4 2005/03/10 06:14:25 deon Exp $

# This script will perform a PG_DUMP to TSM.

if [ -r /etc/sysconfig/tdpLinux.conf ]; then
        . /etc/sysconfig/tdpLinux.conf

else
        #VERIFY="n"
        VERIFY_TMP="/tmp/"
        #VERBOSE="-v"
        LOGFILE="/tmp/tdpLinux.log"

        # PGSQL Settings, make sure your user that you define here has access to all
databases
        # passed via the first command line argument.
        #PGSQL_SERVER="localhost"
        #PGSQL_USER="root"
        #PGSQL_PASSWD="password"

        # The high level file system identifier.

        # Set our path so that all our commands are available.
        PATH=$PATH:/usr/local/bin:/usr/bin:.
fi

# The high level file system identifier.
ADSMPIPE_PATH="/adsmpipe/pgsql"

DB="$1"; shift
RESULT=0
SCRIPT=$(basename $0)

# Skip the first line of mysqldump, it contains the word "Database".
skipFirst=0;

if [ -n "$PGSQL_PASSWD" ]; then
        export PGPASSWORD=$PGSQL_PASSWD
fi

for i in $(psql -l -t ${PGSQL_SERVER:+-h $PGSQL_SERVER} ${PGSQL_USER:+-U $PGSQL_USER}|awk
'{print $1}'); do

        # Skip the first line from the output, it is an SQL header.
        [ $skipFirst -eq 1 ] && skipFirst=0 && continue;

        echo $i >> /tmp/$SCRIPT.$$

        # Ignore template0, it doesnt accept connections.
        [ "template0" == "$i" ] && continue;

        # If we supplied a database on the command line, than just back that one up.
        [ -n "$DB" -a "$DB" != "$i" ] && continue;

        [ -n "$VERBOSE" ] && echo "BACKING up database [$i] $(date)" >> $LOGFILE

        DB_CMD="pg_dump -C ${PGSQL_SERVER:+-h $PGSQL_SERVER} ${PGSQL_USER:+-U $PGSQL_USER}
$i"
```

```
        if [ -n "$VERIFY" ]; then
                $DB_CMD 2>> $LOGFILE | tee /${VERIFY_TMP}/$i.tsm | adsmpipe -f ${i} -c -s
${ADSMPIPE_PATH} ${VERBOSE:-} >> $LOGFILE
2>&1
        else
               $DB_CMD 2>> $LOGFILE | adsmpipe -f ${i} -c -s ${ADSMPIPE_PATH} ${VERBOSE:-}
>> $LOGFILE 2>&1
        fi

        RETURNCODE=$?

        if [ -n "$VERIFY" -a $RETURNCODE -eq 0 ]; then
                echo "- Verify DB checksum [$(md5sum /${VERIFY_TMP}/$i.tsm|awk '{print
$1}')]" >> $LOGFILE 2>&1
                echo "- Verify TSM checksump [$(adsmpipe -f ${i} -x -s ${ADSMPIPE_PATH} |
md5sum|awk '{print $1}')]" >> $LOGFILE 2>&
1
                rm -f /${VERIFY_TMP}/$i.tsm
        fi

        [ -n "$VERBOSE" ] && echo "= Backing COMPLETED with code [$RETURNCODE] $(date)" >>
$LOGFILE

        # If there was an error, then report it
        [ $RETURNCODE -gt 0 ] && echo "! ERROR [${RETURNCODE}] while backing up database
[${i}]" >> $LOGFILE && RESULT=1

done

# Now list all our backups, and see which ones we need to expire.
for i in $(adsmpipe -f \* -t -s ${ADSMPIPE_PATH} 2>&1|awk '{if ($2 == "(A)") {print $5}}');
do

        # If we supplied a database on the command line, than just work on that one.
        [ -n "$DB" -a "$DB" != "$i" ] && continue;

        # Chop off the first slash
        i=${i#/}

        [ -n "$VERBOSE" ] && echo "- Checking DB [$i]" >> $LOGFILE

        if grep -q $i /tmp/$SCRIPT.$$ ; then
                [ -n "$VERBOSE" ] && echo "- DB [$i] is still current." >> $LOGFILE
        else
                [ -n "$VERBOSE" ] && echo "- DB [$i] is no longer current - expiring..." >>
$LOGFILE
                adsmpipe -f $i -d -s ${ADSMPIPE_PATH} ${VERBOSE:-} >> $LOGFILE 2>&1
        fi

done

rm -f /tmp/$SCRIPT.$$
exit $RESULT
```

Restoring PostgreSQL

You must consider how the data was backed up when you are restoring PostgreSQL. The **pg_dump** and **pg_dumpall** commands have options for:

- Backing up all databases in one dump or in an individual database
- Including `create database` in the SQL statements in the dump
- Choosing which format (SQL, **`tar`** or **`pg_restore`**) to use for creating the dump

Once you have determined how the data was sent to TSM, the restore process reverses the backup:

```
adsmpipe -f /pgsqldb -x | psql [-U <username>] [-h <pgsqlhost>] template1
```

If you used the script in Example 7 on page 8 to back up a PostgreSQL database, then you can use the sample script in Example 8 to restore it.

Example 8 Sample of the full script used to restore PostgreSQL databases

```
#!/bin/sh

#$Header: /san/CVS/tdpLinux/pgsql.restore.sh,v 1.3 2005/03/10 06:40:35 deon Exp $

# This script will restore a PGSQL database from TSM.
# The syntax is $SCRIPT database [-i]
# Where -i will return the last inactive version

if [ -r /etc/sysconfig/tdpLinux.conf ]; then
        . /etc/sysconfig/tdpLinux.conf

else
        #VERIFY="n"
        VERIFY_TMP="/tmp/"
        #VERBOSE="-v"
        LOGFILE="/tmp/tdpLinux.log"

        # PGSQL Settings, make sure your user that you define here has access to all
databases
        # passed via the first command line argument.
        #PGSQL_SERVER="localhost"
        #PGSQL_USER="root"
        #PGSQL_PASSWD="password"

        # Set our path so that all our commands are available.
        PATH=$PATH:/usr/local/bin:/usr/bin:.
fi

# The high level file system identifier.
ADSMPIPE_PATH="/adsmpipe/pgsql"

if [ -n "$PGSQL_PASSWD" ]; then
        export PGPASSWORD=$PGSQL_PASSWD
fi

DB="$1"; shift

# Work out if we want Active or Inactive version
if [ "$1" == "-i" ]; then
        TYPE="I"

elif [ -n "$1" ]; then
        echo "Unknown option [$1]"
        exit 1
else
        TYPE="A"
fi
```

```
RESULT=0

for i in $(adsmpipe -t -f /\* -s $ADSMPIPE_PATH 2>&1 | awk '{if ($2 == "("TYPE")") {print
$5}}' TYPE=$TYPE); do

        i=${i#/}

        # If we supplied a database on the command line, than just back that one up.
        [ -n "$DB" -a "$DB" != "$i" ] && continue;

        [ -n "$VERBOSE" ] && echo "RESTORING database [$i] $(date)" >> $LOGFILE

        DB_CMD="psql ${PGSQL_SERVER:+-h $PGSQL_SERVER} ${PGSQL_USER:+-U $PGSQL_USER}
template1"

        adsmpipe -f ${i} -x -s ${ADSMPIPE_PATH} ${VERBOSE:-} 2>> $LOGFILE | $DB_CMD 2>>
$LOGFILE 2>&1

        RETURNCODE=$?

        echo "= Restore COMPLETED with code [$RETURNCODE]"
        [ -n "$VERBOSE" ] && echo "= Restore COMPLETED with code [$RETURNCODE] $(date)" >>
$LOGFILE

        # If there was an error, then report it
        [ $RETURNCODE -gt 0 ] && echo "! ERROR [${RETURNCODE}] while restoring database
[${i}]" >> $LOGFILE && RESULT=1
done

exit $RESULT
```

Backing up OpenLDAP

OpenLDAP has a tool called **slapcat** that exports LDAP databases in LDIF format. The type of database files that your version of OpenLDAP uses determines whether you must stop OpenLDAP before you run slapcat. If you use .dbb files, you can run slapcat while the LDAP server is running. If you use .gdbm files, then you must stop OpenLDAP first.

If you are unsure of your file formats, you may run **slapcat** (as the root user) first. If you receive the following message, stop OpenLDAP before performing the backup:

```
[root@router ldap]# slapcat
slapcat: could not open database.
```

To back up OpenLDAP, shut it down first, if necessary, and use this command:

```
slapcat | adsmpipe -f /ldif -c [-s /adsmpipe/opendldap]
```

Restoring OpenLDAP

OpenLDAP has a tool called **slapadd** that restores OpenLDAP by reading an LDIF file and creating an OpenLDAP database. To restore OpenLDAP:

1. Stop OpenLDAP if it is currently running.
2. Remove the database files that are in the OpenLDAP database directory. The path to the directory is defined in your slapd.conf file. For Red Hat systems, /var/lib/ldap is commonly used. An example of how to remove these files is shown in Example 9.

Example 9 Remove database files from OpenLDAP database directory

```
[root@tdeclt-1 root]# grep ^directory /etc/openldap/slapd.conf
directory       /var/lib/ldap
[root@tdeclt-1 root]# cd /var/lib/ldap
[root@tdeclt-1 ldap]# rm -rf *dbb *gdbm
[root@tdeclt-1 ldap]# ls -al
total 0
drwx------   2 ldap ldap  48 Sep 17 17:01 .
drwxr-xr-x  15 root root 432 Sep 17 17:01 ..
[root@tdeclt-1 ldap]#
```

3. Restore the OpenLDAP database with the following command:

   ```
   adsmpipe -f /ldif -x [-s /adsmpipe/openldap] | slapadd
   ```

4. Change the ownership of the newly created database files so that the OpenLDAP server can read them.

   If your OpenLDAP server does not run as the root (for example, on Red Hat and Fedora systems, it runs with ldap as the user and ldap as the group), then you must change the ownership of the LDAP database files created by the restore process to the user for your slapd process (the same user and group as the owner of the LDAP database directory).

   To determine the user, group, or both that your LDAP runs as, use the commands shown in Example 10.

Example 10 Check user or group that is running LDAP

```
[root@tdeclt-1 root]# grep ^directory /etc/openldap/slapd.conf
directory /var/lib/ldap
[root@tdeclt-1 root]# ls -ald /var/lib/ldap
drwx------  3 ldap ldap 480 Sep  3 13:57 /var/lib/ldap
```

   The output shows that the ldap user is `ldap` and the ldap group is also `ldap`.

   Change the owner and group using the **chown** command:

   ```
   [root@tdeclt-1 root]# chown -R ldap:ldap /var/lib/ldap
   ```

5. Restart your OpenLDAP server.

Building an RPM for adsmpipe

Many Linux distributions use RPM to package applications into a simple file (like a zipped file). The packaging makes it easy to deploy the application and can include (with the application itself):

- Documentation
- Instructions and steps for an automatic start
- Instructions and steps for an automatic stop
- Instructions for installation and removal
- Pre-installation pre-requisite checks

You can use the **rpm-build** command to create your own packages. For more information, see the man pages for **rpm-build**. The .spec file describes how to build the package, and if necessary, how to compile the application.

You must build the adsmpipe utility for each Linux system where the version of glibc or the TSM API is different. If you are familiar with the **rpm-build** command, you can use the .spec file shown in Example 11, which will make it easier to build the utility for each system.

Example 11 Sample .spec file for building adsmpipe with rpm-build

```
%define realname adsmpipe
%define version 1995
%define release 0
%define ext tar.Z
%define TSMAPI %(rpm -q TIVsm-API | sed -e s/^TIVsm-API-// | tr '-' '_')

Summary:                ADSM client for PIPEs
Name:                   %{realname}
Version:                %{version}_%{TSMAPI}
Release:                %{release}
License:                IPL
Group:                  TSM/Clients
Buildroot:              %{_tmppath}/%{name}-buildroot
URL:                    www.redbooks.ibm.com

Source:                 ftp://www.redbooks.ibm.com/redbooks/REDP3980/adsmpipe.tar.Z
Patch:                  adsmpipe.patch

Packager:               Deon George <dgeorge@au.ibm.com>
BuildRequires:          TIVsm-API

%description
This is a STDIN (pipe) client for TSM, enabling you to pipe data that can be stored on your
TSM server as either backup data, or archive data (thus following your management class
policy).

%prep
%setup -q -c %{realname}
mv aix linux
rm -rf solaris
%patch -p 1
echo "TSMAPI is " %{TSMAPI}
%build
cd linux/adsmpipe
make -s

%install
rm -rf $RPM_BUILD_ROOT

mkdir -p ${RPM_BUILD_ROOT}/usr/bin
cp linux/adsmpipe/adsmpipe ${RPM_BUILD_ROOT}%{_bindir}

%clean
rm -rf $RPM_BUILD_ROOT
rm -rf $RPM_BUILD_DIR/%{realname}

%post
echo "This is a TSM API client, dont forget to configure your dsm.sys/dsm.opt files in your
API client directory."

%files
%defattr(-,root,root)
%doc linux/adsmpipe/README
%defattr(0755,root,root)
%{_bindir}
```

This .spec file uses the patch file shown in Example 12 (adsmpipe.patch) to help with the RPM build process.

Example 12 Patch file for building adsmpipe

```
--- ./linux/adsmpipe/Makefile.orig      1995-12-16 09:28:31.000000000 +1100
+++ ./linux/adsmpipe/Makefile   2004-09-17 09:53:13.000000000 +1000
@@ -4,7 +4,9 @@
 #
 #

-CFLAGS=-g
+INCLUDEPATH=`dirname $$(rpm -ql TIVsm-API |grep dsmrc.h)`
+
+CFLAGS=-g -I$(INCLUDEPATH)
 LPATH=.

 FILES=adsmpipe.c adsmblib.c
```

The team that wrote this Redpaper

Deon George is a Senior Pre-Sales Technical Specialist for the IBM Tivoli Storage and TotalStorage® software products, based in Melbourne, Australia. He has been using Linux for 13 years and TSM for 5 years. He focuses on helping customers and colleagues successfully use Linux as an alternative. He is an author of *Linux on IBM* IBM @server *zSeries and S/390: Performance Measurement and Tuning,* SG24-6026, and *Linux on IBM* IBM @server *zSeries and S/390: System Management,* SG24-6820.

Thanks to the following people for their contributions to this paper:

Charlotte Brooks
International Technical Support Organization (ITSO), Poughkeepsie Center

Forsyth Alexander
ITSO, Raleigh Center

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:
This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Send us your comments in one of the following ways:
- Use the online **Contact us** review redbook form found at:
  ibm.com/redbooks
- Send your comments in an email to:
  redbook@us.ibm.com
- Mail your comments to:

IBM Corporation, International Technical Support Organization
Dept. QXXE Building 80-E2
650 Harry Road
San Jose, California 95120-6099 U.S.A.

Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

@server®	zSeries®	Rational®
@server®	AIX®	S/390®
Redbooks ™	DB2®	Tivoli®
Redbooks™	ETE™	TotalStorage®
Redbooks (logo)™	IBM®	
ibm.com®	Lotus®	

Other company, product, and service names may be trademarks or service marks of others.