Section 6
BACKUP AND RECOVERY

In this section: See page

Summary

This section describes backup and reorganization of disk files, the preservation of the execution environment during a power failure, the recovery of files at the record level, the restoration of corrupted files, and the recovery and restart of task groups in an absentee processing environment.

INTRODUCTION

The system provides facilities that enable you to backup and reorganize disk files, preserve the execution environment during a power failure, perform file recovery at the record level, and restart a program from a previously established point.

Backup and Reorganize

The file backup and reorganization facility consists of the save and restore functions. These functions provide you with a means of:

- Backing up disk files or volumes on 1/2-inch or 1/4-inch magnetic tapes, or other media, so that a copy of the files is available should the original files become corrupted.
- Consolidating (concatenating) files that have become fragmented through continuous updating.

Power Resumption

The power resumption facility uses the memory save and auto-restart unit to preserve the memory image through a power failure lasting up to 2 hours. If power is restored during this time, the power resumption facility reconnects the previously online peripheral and communication devices and restarts the tasks that were running when the power failure occurred. If the power failure lasts more than 2 hours, the memory image is destroyed and the power resumption facility disabled. When power is restored, you can reinitialize the system and use the file recovery and checkpoint facilities to restart the system from a previously established restart point.

File Recovery

File recovery enables you to dynamically save record images before they are updated and, if necessary, later write the images back to the file, thereby returning the file to its unaltered state. File recovery provides file integrity in the event of a system failure. The system supports file recovery through three distinct functions:

- Before-image recording - Preserves a record prior to its being updated.
- Cleanpoint or checkpoint declarations - Issued in your program to define a point at which a multirecord or multifile update transaction is complete. When an update transaction is complete, the associated before images are destroyed.
- Rollback, recovery, or restart functions - Returns the files to their unaltered state by applying all before images that have been recorded since the last cleanpoint.

File Restoration

File restoration procedures enable you to reconstruct disk files and/or volumes that are damaged as a result of a device failure. File restoration is provided through two distinct functions:

- After-image recording - Preserves a record of the updates made to files.
- Roll Forward utility - Reapplies updates (after images) to files to bring them up to their most recent consistent state before the device failure.

After images are used in conjunction with the Save, Restore, and Roll Forward utilities to return files to a known state if data in the files is destroyed as a result of a device failure.

The cleanpoint, rollback, and recovery functions should be used to provide file recovery in a transaction-oriented environment. They are best suited for applications in which a single transaction causes a number of record updates. In an absentee processing environment, the checkpoint and restart procedures should be used for file recovery and program restart.

Checkpoint Restart

The checkpoint restart facility enables you to establish a point in your program to which you can return at a later time and continue processing. The return point (checkpoint) is used to save the current status of the task group. You issue a checkpoint call in your program when you reach a point in your processing at which the program could be restarted. A restart can be performed at the most recently completed checkpoint at any time during processing. If the task group is abnormally terminated for any reason, it can be restarted at the most recent valid checkpoint.

FILE BACKUP AND REORGANIZATION

File backup and reorganization is implemented through the Save and Restore utilities. The Save utility transfers disk files and directories to 1/2-inch or 1/4-inch magnetic tape, or to another specified storage medium. The Restore utility reconstructs the saved files and directories and puts them back on disk. Any file that has been saved and restored is automatically reorganized for disk space efficiency.

Since file access time efficiency may be lessened after a file has been in use for some time, it is recommended that disk volumes be periodically saved and restored. The files on the restored volume will be be compacted, resulting in optimal space allocation and improvements in the time required to search directories and check access rights.

Saving Files and Directories

The Save utility enables you to save an entire disk volume, a directory and all its subdirectories and files, or a specified file. If you are saving a directory, you can specify the number of levels of subdirectories (with their associated files) to be saved. Any access control lists associated with the saved files and directories are also saved, unless you specify otherwise.

The saved data, whether a whole volume, a file, or directories and files, is stored in a save file. The save file can be a magnetic tape or disk file, or an output device such as a card punch. When the Save utility processes the files and directories to be saved, it adds information that is meaningful only to the Restore utility. The saved files and directories are not just copies of the originals.

The Save utility can be executed while the files being saved are in use. Used with a journal file (refer to "File Restoration" later in this section), this type of save operation provides a dynamic and concurrent backup facility for high volume systems that cannot afford periodic shutdown to perform static file saves.

Restoring Files and Directories

You can restore from a save file all or part of the data you saved on that file. You can restore an entire volume (if you saved an entire volume), a directory and its associated subdirectories, or a specified file. Whatever you restore, you can return to the place from which you saved it, or you can place it in another directory or another volume.

Data saved from one type of disk can be restored to another type, provided the new disk has the required capacity. For example, you can restore a diskette volume onto a cartridge disk volume, or a partially filled mass storage device volume to a cartridge module disk volume.

Saving Files and Directories

The Save utility enables you to save an entire disk volu directory and all its subdirectories and files, or a spe file. If you are saving a directory, you can specify th of levels of subdirectories (with their associated files saved. Any access control lists associated with the sav and directories are also saved, unless you specify other

The saved data, whether a whole volume, a file, or direc and files, is stored in a save file. The save file can magnetic tape or disk file, or an output device such as punch. When the Save utility processes the files and directories to be saved, it adds information that is mea only to the Restore utility. The saved files and direct are not just copies of the originals.

The Save utility can be executed while the files being s in use. Used with a journal file (refer to "File Restor later in this section), this type of save operation prov dynamic and concurrent backup facility for high volume s that cannot afford periodic shutdown to perform static f saves.

Restoring Files and Directories

You can restore from a save file all or part of the data saved on that file. You can restore an entire volume (i saved an entire volume), a directory and its associated subdirectories, or a specified file. Whatever you resto can return to the place from which you saved it, or you place it in another directory or another volume.

Data saved from one type of disk can be restored to anot type, provided the new disk has the required capacity. example, you can restore a diskette volume onto a cartri volume, or a partially filled mass storage device volume cartridge module disk volume.

: RESUMPTION

Power resumption is an optional facility that allows the system execution environment to be automatically restarted after a power interruption. The central processor must have the memory save and autorestart unit. This unit can preserve the memory image through a power failure lasting up to 2 hours. (It cannot, however, preserve the state of the I/O controllers, nor can it ensure that no operational changes have been made to the mounted volumes.)

If fewer than 2 hours have elapsed when power is returned to the central processor, the power resumption facility will perform the following functions:

- Reinitialize the system software.
- Reconnect peripheral devices.
- Reconnect communication devices serviced by the Asynchronous Terminal Driver (ATD) line protocol handler or the Teleprinter (TTY*) line protocol handler. (Refer to the *System Building and Administration* manual and the *System Programmer's Guide - Volume I* for information about these line protocol handlers.)
- Restart certain application tasks that were active at the time of the failure. Application tasks that are capable of being restarted are those using the display formatting and control facility and those containing user-written code to handle power failure and power resumption.

ıting the Power Resumption Facility

The power resumption facility must be included in the Executive at system building. The central processor must contain a memory save and autorestart unit that has been activated by the operator (refer to the *System User's Guide* for activation procedures).

When power resumption is specified in the system building dialog, all peripheral devices and all communication devices associated with the ATD and TTY* line protocol handlers are designated as reconnectable and will be automatically reconnected when power is restored. If any ADT or TTY*-associated device is not to be automatically reconnected, the Set Terminal Characteristics (STTY) directive associated with the device must not contain the -RECONNECT argument.

Power Resumption Functions

The power resumption facility automatically performs th following functions:

- Restarts the device drivers, clock, communications subsystem, and display formatting and control facil
- Reconnects all peripheral devices that were online a time of the failure.
- Reconnects ATD- or TTY*-associated communication dev that were online at the time of the failure, except those devices designated as not reconnectable.
- Restores the screen forms on reconnected terminals controlled by the display formatting and control fac
- Resets the system date and time if the date/time clc separate battery backup unit.
- Reloads the memory management unit.
- Reestablishes the integrity of mounted volumes.
- Restarts application tasks that were active when the failure occurred, provided the tasks used the displa formatting and control facility or contained user-wr code to handle power failure and power resumption.

If an application task is to be notified when a power re has occurred, it must be written to check Trap 53 when t becomes active and is issuing its own instructions (not executing Executive instructions). (Refer to "Trap Hand Section 5.)

After a power resumption has occurred, peripheral device reconnectable ATD- or TTY*-associated devices that were at the time of the failure are again brought online. Th operator may be required to initialize certain periphera devices. A terminal user may be required to reenter the line if he/she had not pressed the RETURN or XMIT key wh failure occurred. (Refer to the *System User's Guide* for details.)

ꞆOVERY

The file recovery facility enables you to save record images from a file before it is updated and to later write these images back to the file, eliminating the alterations made during the updating. Every time a record is updated, a copy of the record, as it exists before the update, is written to a system-created file. The system-created file is called a recovery file; the records it contains are called before-images.

The system uses recovery files to bring your data files to a consistent state following a software failure or a system failure such as that caused by a loss of power. When the before-images are applied in reverse chronological order to your data files, the data files are rolled back to a previously established state.

ting Recoverable Files

File recovery is optional. You designate a file as recoverable through the -RECOVER argument of the Create File command. Files not created as recoverable can be made recoverable by specification of the -RECOVER argument of the Modify File Attribute (MFA) command. Also, you can designate all files in a directory or volume to be recoverable through the Modify Directory Attributes (MDA) command.

Recoverable files can be made nonrecoverable through the specification of the -NORECOVER argument in the MFA or MDA command.

y File Creation

Each task group (or task, in some cases) having a data file designated as recoverable has associated with it a recovery file. The recovery file is created by the system when the first before-image for a recoverable file is about to be written.

All recovery files are created subordinate to your working directory, unless you specified otherwise by the Assign Recovery File command. (The names of the files are recorded in the $$CATALOG directory, which is positioned under the root directory of the system volume. This directory is maintained by the system.) Each recovery file is assigned a name of the form:

```
$$REC.nnggtt
```

where nn is the node identifier, gg is the group identifier, and tt is the task identifier.

File Recovery Process

The system recovers a data file (erases the updates made by writing the before-images back into the file.

You declare points in your task group processing (calle cleanpoints) at which all file updates are considered va When a cleanpoint is declared, all before-images taken u that point are invalidated. New before-images are writt you again begin to update the file.

You can perform a rollback at any time during processin a rollback is requested, the before images are written i file, wiping out updates made since the last cleanpoint.

Use of the cleanpoint and rollback task group functions recommended in a transaction-oriented environment.

Taking Cleanpoints

When you consider the data in your task group's file(s) consistent and valid, declare a cleanpoint in your task Cleanpoints are established by $CLPNT macrocalls in Asse language programs and by the ZCLEAN utility in programs in higher-level languages (for example CALL "ZCLEAN" in When a cleanpoint is declared, the system performs the f actions:

- Writes all buffers modified by the task group to dis
- Updates all directory records for files modified by group.
- Invalidates the recovery file before-images that hav taken for data files used by the task group.
- Unlocks all previously locked records belonging to t group. (Tasks waiting for these records are activat

The File System automatically performs a cleanpoint when recoverable file is closed.

g Rollback

Rollback initiates the recovery of your task group's files to the condition in which they were at the last cleanpoint. If programming in Assembly language, request a rollback by coding a $ROLBK macrocall. If programming in a higher-level language, request a rollback by using the ZCROLL utility (for example, in COBOL use a CALL "ZCROLL" statement).

When you request a rollback, the system performs the following actions:

- Takes before-images from the recovery file and writes them into the data files used by the task group, thereby wiping out updates made since the last cleanpoint.
- Invalidates the before-images for the task group's data files on the recovery file.
- Unlocks all previously locked records belonging to the task group. (Tasks waiting for these records are activated.)

The File System automatically performs a rollback when a task group terminates abnormally.

j After System Failure

If recovery files exist, the operator should issue the Recover command so that the system will perform a rollback of all recoverable data files.

TORATION

File restoration provides the ability to preserve updates that have been made to files, and to apply these updates to saved versions of the files if the original versions become corrupted. You cause images of records that have been modified (after-images) to be recorded in a journal (after-image) file. You can then use the journal file in conjunction with the Save, Restore, and Roll Forward commands to restore files to a known state if data in the files is destroyed as a result of a device failure. (If I/O errors indicate any damaged files and/or volumes, file restoration procedures are recommended).

Designating Restorable Files

You designate files as restorable by specifying the -RES argument of the Create File command. Files not created restorable can be made restorable by specifying the -RES argument of the Modify File Attribute (MFA) command. Al can designate all files in a directory or volume to be restorable through the Modify Directory Attributes (MDA) command.

It is recommended that files designated as restorable al designated as recoverable (having the -RECOVER attribute provide for complete file integrity if a device or syste failure occurs.

Restorable files can be made nonrestorable by specifying -NORESTORE argument of the MFA or MDA command.

Journal File Creation

The journal file is created and maintained by the operat through the Open Journal, Close Journal, Display Journal Swap Journal commands. One system-wide journal file, on disk, records updates made to all restorable disk files. system-created journal history file contains the name of current journal file and a history of all previous journ files, including the date and time they were created.

Each time a record in a restorable file is updated, the records on the journal file the image of the record as i after the modification (the after-image). The after-ima the updated record is written to the journal file at the the record in the file is physically updated. If the op specifies the -BEFORE argument in the Open Journal comma system will also record on the journal file the image of record as it exists before the modification (the before- You might want to record before-images for audit purpose

The journal file contains a running summary of all chang to restorable files (for example, if a restorable file i renamed or modified, appropriate entries are added to th journal file to reflect these changes). Restorable disk cannot be modified in any way unless the journal file ha previously opened by the operator.

toration Process

For each file that is corrupted, the restoration process involves mounting a known valid version of the file, reconstructed from data preserved during a previous save operation. The save operation involves preserving the data contents and selected attributes of the uncorrupted file (by means of the Save command) before any catastrophe occurs, then restoring the file structures of the saved file (using the Restore command) after the file has been corrupted. Following these actions, you cause after-images from the journal file to be applied to the restored file by using the Roll Forward command. The restored file now incorporates the changes or updates stored in the journal file since you last invoked the Save command.

File restoration offers more extensive procedures if files are corrupted following a device failure and file recovery procedures fail to return files to a consistent state.

For example, the operator opens the journal file and enters the Recover command. If the Recover command executes successfully, you can log in and continue processing. If the Recover command fails to execute successfully, the operator must close the journal file, mount saved versions of all files, and use the Restore command to reinstall the saved versions. The Roll Forward command is then entered. This command applies journal file images to all restored files, thereby updating the files to reflect modifications made after Save commands were entered for those files. File restoration is then complete and users can log in and continue processing.

OINT RESTÁRT

The checkpoint restart facility allows you to provide a task group file recovery and program restart capability in an absentee processing environment. Through checkpoint restart, you can establish a point in your program to which you can return at any time and continue processing. This return point (called a checkpoint) is used to save the current status of the task group request.

You can perform a restart to the most recently completed checkpoint after the abnormal termination of the task group request or at any point during the processing of the task group request. A restart cannot be performed from an earlier checkpoint, nor can it be performed after the normal termination of a task group request.

Checkpoint restart does not support the use of the Listener secondary login facility.

Designating Restorable Files

You designate files as restorable by specifying the -RES argument of the Create File command. Files not created restorable can be made restorable by specifying the -RES argument of the Modify File Attribute (MFA) command. Al can designate all files in a directory or volume to be restorable through the Modify Directory Attributes (MDA) command.

It is recommended that files designated as restorable al designated as recoverable (having the -RECOVER attribute provide for complete file integrity if a device or syste failure occurs.

Restorable files can be made nonrestorable by specifying -NORESTORE argument of the MFA or MDA command.

Journal File Creation

The journal file is created and maintained by the operat through the Open Journal, Close Journal, Display Journal Swap Journal commands. One system-wide journal file, on disk, records updates made to all restorable disk files. system-created journal history file contains the name of current journal file and a history of all previous journ files, including the date and time they were created.

Each time a record in a restorable file is updated, the records on the journal file the image of the record as i after the modification (the after-image). The after-ima the updated record is written to the journal file at the the record in the file is physically updated. If the op specifies the -BEFORE argument in the Open Journal comma system will also record on the journal file the image of record as it exists before the modification (the before- You might want to record before-images for audit purpose

The journal file contains a running summary of all chang to restorable files (for example, if a restorable file i renamed or modified, appropriate entries are added to th journal file to reflect these changes). Restorable disk cannot be modified in any way unless the journal file ha previously opened by the operator.

toration Process

For each file that is corrupted, the restoration process involves mounting a known valid version of the file, reconstructed from data preserved during a previous save operation. The save operation involves preserving the data contents and selected attributes of the uncorrupted file (by means of the Save command) before any catastrophe occurs, then restoring the file structures of the saved file (using the Restore command) after the file has been corrupted. Following these actions, you cause after-images from the journal file to be applied to the restored file by using the Roll Forward command. The restored file now incorporates the changes or updates stored in the journal file since you last invoked the Save command.

File restoration offers more extensive procedures if files are corrupted following a device failure and file recovery procedures fail to return files to a consistent state.

For example, the operator opens the journal file and enters the Recover command. If the Recover command executes successfully, you can log in and continue processing. If the Recover command fails to execute successfully, the operator must close the journal file, mount saved versions of all files, and use the Restore command to reinstall the saved versions. The Roll Forward command is then entered. This command applies journal file images to all restored files, thereby updating the files to reflect modifications made after Save commands were entered for those files. File restoration is then complete and users can log in and continue processing.

OINT RESTÁRT

The checkpoint restart facility allows you to provide a task group file recovery and program restart capability in an absentee processing environment. Through checkpoint restart, you can establish a point in your program to which you can return at any time and continue processing. This return point (called a checkpoint) is used to save the current status of the task group request.

You can perform a restart to the most recently completed checkpoint after the abnormal termination of the task group request or at any point during the processing of the task group request. A restart cannot be performed from an earlier checkpoint, nor can it be performed after the normal termination of a task group request.

Checkpoint restart does not support the use of the Listener secondary login facility.

Checkpoint

When a task requests a checkpoint, the system records tł
current contents of the task group's memory and the curr
state of tasks, files, and screen forms onto a checkpoir
you have previously assigned. The system then takes a
cleanpoint for that task group so that recoverable files
synchronized with that checkpoint. (Refer to "File Recc
earlier in this section for a description of recoverable
and cleanpoints.)

The system supports one checkpoint task and any number c
tasks that are dormant or waiting on requests placed aga
other tasks in the task group. (Thus, a single active c
executing under the Command Processor and/or any number
nested EC files can be checkpointed.)

Checkpoint File Assignment

You enable the checkpoint restart facility for your task
and designate where its checkpoint images are to be recc
issuing the Checkpoint File Assignment command.

Checkpoints are written alternately into each of a pair
checkpoint files. This technique ensures the availabili
the previous valid checkpoint if a failure occurs during
process of taking a checkpoint. The system locates and
only the most recently completed successful checkpoint f
pair of checkpoint files that you specified.

Pathname

When designating the checkpoint file, specify a single p
(the last element of which can be a maximum of 10 charac
The system appends the suffixes .1 and .2 as appropriate
the system cannot find one or both of the specified chec
files, it creates it (them).

Checkpoint

When a checkpoint is taken, the system writes a checkpoint image and performs a cleanpoint for all recoverable files being used by the task group. If programming in Assembly language, request a checkpoint by coding a $CKPT macrocall. If programming in a higher-level language, request a checkpoint through the ZXCKPT utility (in COBOL you can use a CALL "ZXCKPT" statement or the RERUN clause in the I-O-CONTROL paragraph).

nt-
te

Your task group must be in a state that allows the system to take checkpoints. To be in a checkpointable state, tasks in the group can have no outstanding requests against tasks outside the group. Specifically, a task group is in a checkpointable state when each task that is part of the group has requested a checkpoint, is waiting on a request issued to another task in the task group, or is dormant (no current requests exist for the task).

Once a checkpoint is recorded by a task group, it remains available as a restart point until the next checkpoint request is completed, the current checkpoint file is disassigned (by the -DISASSIGN argument of the Checkpoint File Assignment command), or the task group request is terminated normally.

You can use the Defer Checkpoint macrocall to prevent or allow the taking of checkpoints by tasks within the task group. If you wanted to protect a procedure from being checkpointed, you would disable checkpoints at the beginning of the procedure and enable them at the end of the procedure.

The lead task of the group may be waiting for both another task that is a member of the group and a "break" request.

Checkpoint Processing

When a task group takes a valid checkpoint, the system ı the following information on the current checkpoint fil established for that group:

1. Executive information, including data structures, u: memory blocks obtained by Get Memory operations, da segments of bound units linked with separate code a nonshareable bound units, and floatable overlays.

2. Status and pathnames of the standard I/O files.

3. Memory locations and pathnames of shareable bound u

4. Current state of screen forms for terminals operati the display formatting and control facility.

5. Status and position of all active user files (files have been associated, reserved, or opened)..

When your file information has been recorded, the check image is completed and a cleanpoint is taken. You must that files to be synchronized with the checkpoint resta process have been designated as recoverable. Since the System performs a cleanpoint when a recoverable file is you may have to take a checkpoint prior to closing the f keep checkpoint restart synchronized with the state of t recoverable file. (Temporary files cannot be designate recoverable.)

Checkpoints cannot be taken while an active local mail m group exists. In other words, a checkpoint cannot be ta the period between message initiation or acceptance and termination; rèfer to "Message Facility Macrocall Interf Section 5.

Checkpoints are not made automatically obsolete by the n termination of the task under which they were issued. T invalidate a previous checkpoint (taken during the execu one command) before processing a new command, you must t checkpoint immediately prior to the termination of that t

You can perform a restart at the following times:

- During the processing of the task group request that issued the checkpoint request.
- During the processing of a task group request scheduled after the abnormal termination of the task group request in which the checkpoint was taken.
- When the system is reinitialized following a system failure.

When a restart request is issued, the task group issuing the request is terminated abnormally and the task group request recorded on the checkpoint file is again put into effect.

The system locates the most recently completed checkpoint and reads the checkpoint image from the file, rebuilding the Executive data structures and memory blocks, reloading bound units, and repositioning active user files.

Procedural code and workspace must occupy the same physical memory locations that were used when the checkpoint was taken. In general, task groups that are to be restarted must be the sole users of memory pools. Shareable bound units referred to by these groups must be permanently loaded (through the Load command in the system startup EC file). The configuration under which the restart is performed must be identical to that which existed when the checkpoint was taken.

Requesting a Restart

To restart from the last completed checkpoint (and to ab
current task group request, if restarting during the se
issue the Restart command. The operator can restart an
task group that has a valid checkpoint by using the -GR
argument of the Restart command. If the memory blocks
to effect the restart are not available, the restart is
aborted. Specification of the -WTMEM argument of the R
command causes the system to wait until the specific me
blocks required to perform the restart become available.

If this is a restart following a system failure, the Re
command must have been issued by the operator, or throu
file, to perform a system-wide rollback of all recovera
files.

If a restart is performed during a session, the abort
(termination) of the group request causes a rollback of
recoverable files in your task group. The abnormal ter
of the group request causes the last completed checkpoi
to be retained as a valid checkpoint. The Abort Group
Group Request commands force an abnormal termination; t
command causes a normal termination. (The normal termi
the Command Processor with a nonzero value in the $R2 r
is treated as an abnormal termination for checkpoint fi
purposes.)

The Validate Checkpoint command or active function can
to ascertain whether the specified checkpoint file pair
a valid restartable checkpoint.

Processing

When you issue the Restart command, the system performs the following steps:

1. Locates the most recently completed checkpoint.

2. Validates that the restart is being performed under the same user id as that used when the checkpoint was taken.

3. Rebuilds Executive data structures.

4. Reads nonshareable bound units, data segments, floatable overlays, and memory blocks that were obtained by get-memory operations from the checkpoint image into the same memory locations they occupied at the time the checkpoint was taken.

5. Reloads shareable bound units in the system memory pool. Only the code segment is reloaded if the bound unit was linked with separate code and data. Unless it was linked with the restart relocatable attribute (Linker RR directive), the code segment is reloaded at the same system pool memory locations occupied when the checkpoint was taken.

6. Associates, gets, opens, and positions active user files recorded on the checkpoint image. Rollback should have been performed already (refer to "Requesting a Restart" above).

7. Restores the screen content of terminals that were operating under the display formatting and control facility and were active at the time of the checkpoint.

8. Reissues the break request if such a request had been issued by the lead task at the time of the checkpoint.

9. Turns on the task that issued the checkpoint request at the next sequential instruction after the checkpoint.

The checkpointed state of the standard I/O files (user-in, user-out, command-in, and error-out) is reestablished at restart time. Modifications made to files (for example, EC files) between the checkpoint and the restart must be restricted to those that do not invalidate the repositioning of the files. A command being restarted must remain in the same position in the file; only those commands that follow the restarted command have any effect on the restarted task group request.

Shareable bound units being used by a checkpointed task are reloaded and not restored from a checkpointed memor (except for the data segments of bound units linked wit separate code and data). Thus, all such bound units sh contain only code. All shareable bound units in use by restarting task group must be identical to the versions existed at the checkpoint. They cannot be relinked. I Overlay Area Table (OAT) is in use for such a bound uni overlay area can be reserved at the time the checkpoint taken.

If you have application programs that issue physical I/ for communication devices, you must reissue connects to devices before issuing Read and Write orders to them.