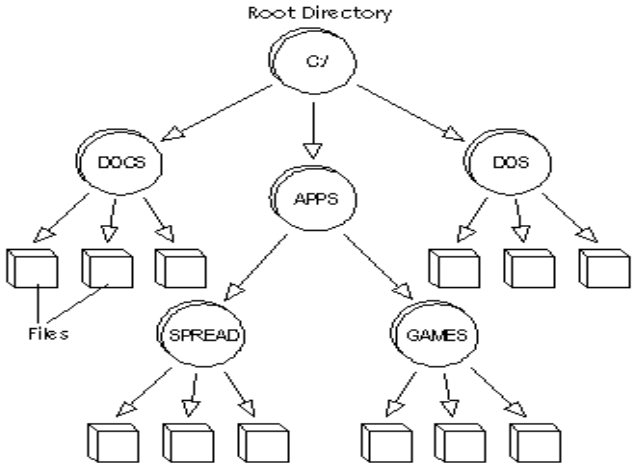

Ab initio Session 8
Introduction to Ab Initio
Capgemini
Ab Initio Training
1

- **Supported Flat File Formats**
- **Layout**
- **Multifile System Concepts**
- **Creation of MFS and Multidirectory**

File System

- The system that an operating system or program uses to organize and keep track of files.
- For example, a *hierarchical file system is one that uses directories to organize files into a tree structure.*



CapGemini **Ab Initio Training** 4

Processing Flat Data Files: Serial Files and Multifiles

This section describes the following:

- About Flat Files
- Supported Flat File Formats
- Locating Files with URLs
- Setting File Permissions

Flat File

- A flat file is a file containing records, generally one record per line.
- Fields may simply have a fixed width with padding, or delimited by white space or tabs or commas or other characters.
- Extra formatting may be needed to distinguish an internal space from a delimiter.
- There are no structural relationships.
- The data are "flat" as in a sheet of paper, in contrast to more complex models such as a relational database.
- A flat file database is a database that stores data in a plain text file.
- **Flat file is also a type of computer file system that stores all data in a single directory.**

Flat File-Example

- The classic example of a flat file database is a basic name-and-address list, where the database consists of a small, fixed number of fields: *Name* ,ID, and *Phone Number* or Project team.

id	name	team
1	Amy	Blues
2	Bob	Reds
3	Chuck	Blues
4	Dick	Blues
5	Ethel	Reds
6	Fred	Blues
7	Gilly	Blues
8	Hank	Reds

Flat File-cont.

- There are no folders or paths used organize the data. While this is a simple way to store files, a flat file system becomes increasingly inefficient as more data is added.
- Some databases, such as Microsoft Access, are flat file databases. They store data in flat files that are external to the database.
- Common flat file types include: ASCII or EBCDIC files, comma-separated lists, excel spreadsheets, and XML files.
- Flat files are files that contain unstructured data.
- You can read data into and out of flat files using the Graphical Design Interface (GDE).

About Flat Files-cont.

- When working with flat files in the GDE, it is important to understand the structure of the data, the number of records, how you want to move them, what constitutes bad data, and so on. The data structure, rather than the file type is of greatest importance to the GDE.

- Flat files can be *non-parallel* or *parallel*. The simplest form is non-parallel where there is a one-to-one relationship for each file. Parallel files, on the other hand, consist of more than one file called partitions.

Supported Flat File Formats

The Ab Initio Graphical Design Interface (GDE) supports these flat file formats:

- Serial Files
- Multifiles
- Ad-hoc Multifile

All file types use the **.dat** extension.

Serial Files

- A serial file is a flat, non-parallel file also known as *one-way parallel*.
- You create serial files using a Universal Resource Locator (URL) on the component's Description tab. The URL starts with **file:** for example, file://node.abinitio.com/dat/input.dat.

Serial Files-cont.

- A **serial file** is one in which the records have been stored in the order in which they have arisen. They have not been sorted into any particular order.
- An **example** of a serial file is an **unsorted transaction file**.
- A **shopping list** is an example of a non-computerized serial file.
- Serial files can be stored on tape, disc or in memory.

Parallel File

- The parallel file consists of an ordered set of standard files, usually on different disks or on different systems.
- The parallel program is one that runs n times from copies of the same executable, resulting in the processes ,which are usually on different CPUs.

File Management

- The Ab Initio Co-Operating System provides facilities for managing large datasets that span many disks or are distributed across multiple servers.
- The Co-Operating System supports **multifile systems**, which allow users to manage a set of distributed files as a single entity. When your data is stored in Ab Initio Multifiles ,you can apply familiar file operations to the whole collection of the files from the central point of administration. Further Ab Initio application can access files anywhere within an enterprise, using the familiar URL.

Multifiles

- A multifile is a parallel file consisting of individual files called partitions and often stored on different disks or computers.
- A multifile is essentially the "global view" of a set of ordinary files, each of which may be located anywhere where the Ab Initio Co>Operating System is installed.
- Each partition of a multifile is an ordinary file.
- By using the global view and multifiles, you can avoid having to draw data parallelism explicitly.
- A multifile has a control file that contains URLs pointing to one or more data files.

Multifiles

- Multifiles are parallel files composed of individual files located (usually, but not necessarily) on different disks or on different systems. The individual files are referred to as the partitions of the multifile.
- Note that the icon for a multifile has 3 platters instead of 2.
- Ab Initio utilities let you copy, rename, delete, etc., multifiles as easily as ordinary files.

Multifiles-cont.

You can divide data across partition files using these methods:

- random or round robin partitioning
- partitioning based on ranges or functions
- replication or broadcast
  in which each partition is an identical copy of the serial data.

Multifiles-cont.

- Multifiles reside in multidirectories.
- Multidirectories and multifiles are identified using URL syntax with "`mfile:`" as the protocol part:
  - A Multidirectory
    `mfile:/users/training-07/test-mfs/`
  - A Multifile within a Multidirectory
    `mfile:/users/training-07/test-mfs/xx.dat`
- These URL's are simply abstractions for the many pieces making up a Multidirectory or multifile.

Ad-hoc Multifile

- An ad-hoc multifile is a also a parallel file.
- Unlike a multifile, however, the content of an ad-hoc multifile is not stored in multiple directories.
- In a custom layout, the partitions are serial files.
- You create an ad-hoc multifile using partitions on the component's Description tab.

Locating Files with URLs

- Ab Initio software uses Universal Resource Locators (URLs) to locate files.
- You enter URLs for datasets, record formats, input and output files, and so on in a component's Properties dialog.
- Enter files and multifies on the Description tab, transforms on the Parameters tab, and DML record formats on the Ports tab.
- The Ab Initio URL format is:

[file | mfile] ://*hostname*/*directory1*/*directory2*.../*filename*

URL Format

- Description

Argument	Description
file | mfile	The protocols are: • file — Specifies a serial file. • mfile — Specifies a multifile.
hostname	Specifies the name of the computer containing the file you want.
directory1 ...	Specifies the directory path to the file.
filename	Specifies the filename.

Examples

- This example specifies a file named **input.dat**, located in the **tmp** directory on a computer named **revkalt.abinitio.com**:

file://revkalt.Ab initio .com/tmp/input.dat

- This examples specifies a multifile named **customer .dat**, located in the **tmp/mfs** subdirectory on a computer named **mycomputer**:

mfile://mycomputer .Ab initio .com/tmp/mfs/customer.dat

Setting File Permissions

You can set permissions to protect output and temporary dataset files. To set permissions, use the Properties dialog: Access tab. The checkboxes on the Properties Access tab match the UNIX file protection standards for user, group, and others.

They are:

- **r** — read access
- **w** — write access
- **x** — execute access

More on Multifiles

- Data parallelism makes it possible for Ab Initio software to process large amounts of data very quickly. In order to take full advantage of the power of data parallelism, you need to store data in its parallel state. In this state, the partitions of the data are typically located on different disks on various machines. An Ab Initio ***multifile*** is a parallel file that allows you to manage all the partitions of the data, no matter where they are located, as a single entity.

Multifile Systems

- You organize and manage Multifiles by using a ***multifile system,*** which has a directory tree structure, allowing you to work with Multifiles and the directory structures that contain them in the same way you would with serial files. Using an Ab Initio multifile system, you can apply familiar file management operations to all the partitions of a multifile — no matter how diverse the machines on which they are located — from a central point of administration by referencing the ***control partition*** of the multifile with a single ***URL*.**

Multifile Systems

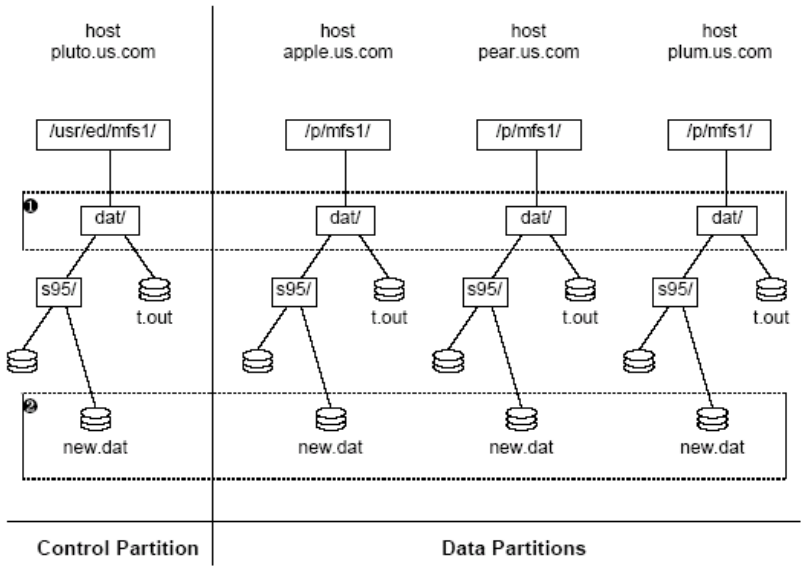
- Ab Initio multifiles reside in parallel directories called multidirectories, which are organized into multifile systems.
- An Ab Initio multifile system consists of multiple replications of a directory tree structure containing multidirectories and multifiles.
- Each replication constitutes a partition of the multifile system. Each partition holds a subset of the data contained in the multifile system, and the system has one additional partition that contains control information.
- The partitions containing data are the *data partitions* of the system, and the additional partition is the *control partition*.
- The control partition contains no user data, only the information the Co>Operating System needs to manage the multifile system.

Multifile Systems

- Visualize a directory tree containing subdirectories and files. Now imagine *n* identical copies of the same tree located on several disks, and number them **0** to *n*-**1** (the Co>Operating System numbers partitions starting at **0**).
- These are the data partitions of the multifile system. Then add one more copy of the tree to serve as the control partition. This is a multifile system. Note that each Multidirectory and multifile in the system has a control partition and data partitions in the corresponding partition of the multifile system.
- You can place the control and data partitions of a multifile system on any computer that has the Co>Operating System installed on it and to which the run host can connect.

A Sample Multifile System

- The diagram below shows multifile system with its control partition at//pluto.us.com/usr/ed/mfs1.The multifile system has three partitions and consequently so do all the Multidirectory and Multifiles it contains. The three partitions of multifile system are at various locations on the disks of three computers: apple.us.com , pear.us.com , plum.us.com.
- The following is the Three Partition multifile system:



CapGemini **Ab Initio Training** 29

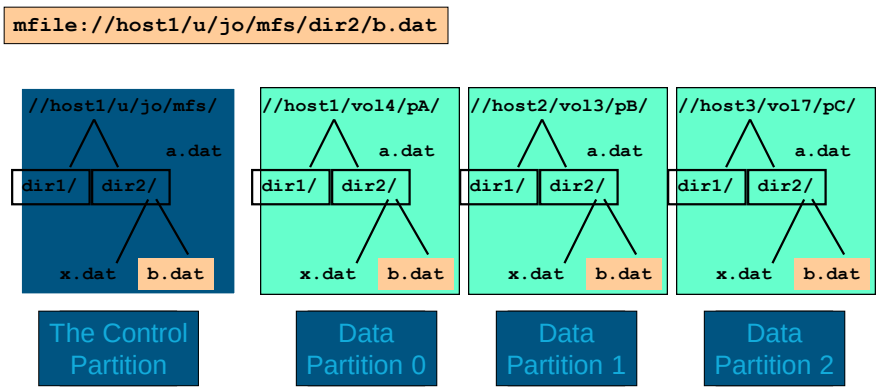
MULTIDIRECTORIES

- Multifiles are stored in parallel directories called multidirectories,which reside in a multifile system.
- An Ab Initio multifile system is a replication of the native operating system's directory tree structure, designed such that each partition can hold the subset of the data to be processed.
- The multifile system has an additional partition that contains control information.

CONTROL FILES

- A Multidirectory consists of a control directory (a directory somewhere in a control partition), and the n corresponding directories from the partitions of the multifile systems .
- Similarly, a multifile consists of a control file (a file somewhere in the control partition), and the n corresponding files from the partitions of the multifile systems.
- There is no user data in the control file.

A Multidirectory Hierarchy



File management utilities

- The Ab Initio file utilities provide the ability to manipulate files and directories, including Multifiles and Multidirectory, regardless of where in the enterprise they are located.
- These utilities are similar in functionality to a group of Unix utilities: the Ab Initio versions' names differ from their Unix counterparts' in having the characters **m_** prefixed. For example, the **m_cp** command functions similarly to the Unix **cp** command.
- The important difference is that with the Ab Initio file utilities, you use URLs to specify file and directory paths. This allows files to be located on remote hosts and to be Multifiles or multidirectories. Some other small semantic differences exist, usually related to error reporting or usage of file protections, and these will be mentioned where appropriate.

Managing Multifiles

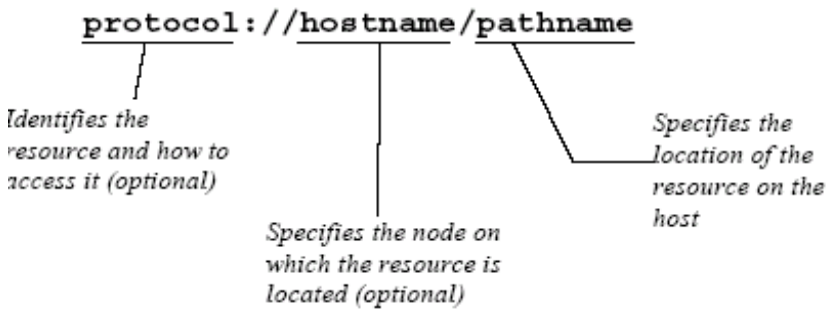
- Ab Initio multifile systems enable users to manage as a single entity large datasets that span many disks or are distributed across multiple computers.
- When you store data in Ab Initio Multifiles, you can apply familiar file operations to the whole collection of files from a central point of administration, using Co>Operating System file shell commands that work like, and are named like, their Unix counterparts. For example, the **m_cp** Co>Operating System command works like the Unix **cp** command.
- Unlike Unix commands, however, Co>Operating System file commands use URLs to reference files, enabling the Co>Operating System to access files on any computer that is running the Co>Operating System and to which the run host can connect.

Referencing Multifiles

- You reference a multifile system by referencing the root of the control partition. You reference Multifiles and multi directories by referencing the files and directories within the control partition.
- URL:The Ab Initio system uses Universal Resource Locator (URLs) to identify distributed resources. Each partition of a layout that is each hostname/pathname pair-is specified by a URL,as used on www.

Referencing Multifiles-URL

- A URL constructed like this:



Referencing Multifiles-URL

In Ab Inito directory and file references:

- The protocol field indicates whether the file is a standard (serial) file or a partition of a multifile.
- The hostname field specifies the computer where the partition resides.
- The pathname is a standard pathname ,in the form accepted by the host's native OS ,indicating the partition's location on the host.

Referencing Multifiles-Protocol Field

The optional protocol field of a URL may contain a file type:

- Mfile: for Multifiles
- File: for standard (serial) files

Hostnames in a URL are internet host names. For example, a computer known as yang.com on the internet or on a disconnected intranet would be referred to by that name in Ab Inito URLs.

Creating a Multifile System

- To create a multifile system, you issue the **m_mkfs** command, using as arguments the URLs of the partitions of the multifile system you want to create. The first URL creates the control partition, and each subsequent URL creates the next partition of the multifile system.
- For example, the following command creates the root level of the multifile system shown in A Multifile System:

Example of m_mkfs command

- Example:
- m_mkfs c :/ddata c:/ndata/m0 c:/ndata/m1

Execute Command Explanation

The URLs in the above command serve the following purposes:

- **c :/ddata:** The first URL specifies the location of the root directory of the control partition of the multifile system and gives the multifile system the name by which the Co>Operating System recognizes it.
- **c:/ndata/m0:** The second URL specifies the location of the root directory of the first data partition (the Co>Operating System numbers partitions starting at **0**).
- **c:/ndata/m1:** The third URL specifies the location of the root directory of the second data partition.

If you added another URL to the command, it would create a multifile system with four data partitions, and so on.

Multifile directories Existence

The multifile system directories named in the **m_mkfs** command must not exist before you issue the command; the command must create them. Their parent directories, however, must exist before you issue the command.

In other words, before you issue the above **m_mkfs** command:

- The **c:** must exist on computer but not /ddata
- The **/ndata** directory must exist on **c:**, but **m0** must not.
- The **/ndata** directory must exist on **c:**, but **m1** must not.

Layout

- Before you can run an Ab Initio graph, you must specify *layouts* to describe the following to the Co>Operating System:
- The location of files
- The number and locations of the partitions of multifiles
- The number of, and the locations in which, the partitions of program components execute

Layout

A layout is one of the following:

- A URL that specifies the location of a serial file
- A URL that specifies the location of the control partition of a multifile
- A list of URLs that specifies the locations of:
  - The partitions of an ad hoc multifile
  - The working directories of a program component

Layout-cont.

- Every component in a graph — both dataset and program components — has a layout. Some graphs use one layout throughout; others use several layouts and repartition data as needed for processing by a greater or lesser number of processors.

Layout-cont.

- During execution, a graph writes various files in the layouts of some or all of the components in it. For example:
- An Intermediate File component writes to disk all the data that passes through it.
- A phase break, checkpoint, or watcher writes to disk, in the layout of the component downstream from it, all the data passing through it.
- A buffered flow writes data to disk, in the layout of the component downstream from it, when its buffers overflow.
- Many program components — Sort is one example — write, then read and remove, temporary files in their layouts.
- A checkpoint in a continuous graph writes files in the layout of every component as it moves through the graph.

Layout-cont.

Critical Concerns

- The layouts you choose can be critical to the success or failure of a graph. In order for a layout to be effective, it must fulfill the following conditions:
- The Co>Operating System must be installed on the computers specified by the layout.
- The run host must be able to connect to the computers specified by the layout.
- The layout must allow enough space for the files the graph needs to write there.
- The permissions in the directories of the layout must allow the graph to write files there. If a layout does not fulfill these conditions, the graph will fail. See Actual Working Directories for details about exactly where the Co>Operating System locates the files it writes during the execution of a graph

Platform-specific Considerations-Multifiles

This section contains the following topics dealing with considerations peculiar to particular platforms:

- Unix
- UNC Syntax
- IBM OS/390
- Hostnames on IBM SP2

Platform-specific Considerations-Multifiles

- When creating multifile systems on Unix platforms, we recommend that you do not specify the path to the partitions of the multifile system in terms of a network-mounted file system, such as NFS, unless the files located on that partition are small. Large files specified in terms of a network mounted file system can degrade graph performance across a network.

Creating a Multidirectory

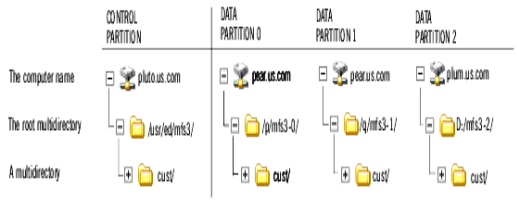
- To continue building the multifile system started in Creating a Multifile System and shown in A multifile system, you need to add the **cust** multidirectory. To create a multidirectory in a multifile system you use the **m_mkdir** command. The command uses only one URL as an argument, the URL for the control partition of the multidirectory you want to create. Given this URL, the **m_mkdir** command creates the control and data partitions of the multidirectory.

m_mkdir Command

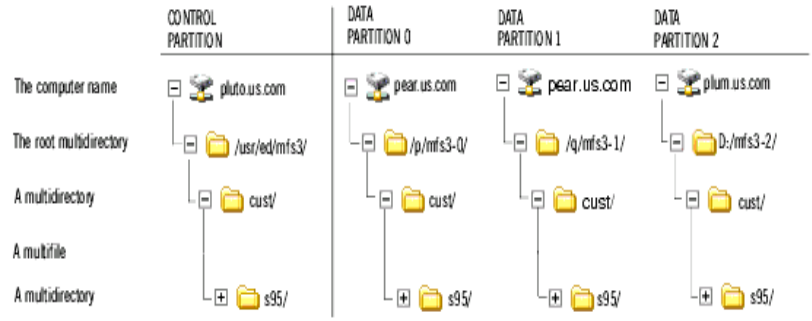
- Following is the command that creates the multidirectory named **cust** in the multifile system mfile:/c:/ddata:
- **m_mkdir //c:/ddata /cust**

The tree of the multifile system now looks like this:

- To create another multidirectory, this one a subdirectory named **s95** in **cust**, issue the following command:
- **m_mkdir //c:/ddata /cust/s95**



The tree of the multifile system now looks like this:



More on Creation of Multidirectory

- As with the **m_mkfs** command, the multidirectory you are creating must not exist before you issue the **m_mkdir** command; the command must create it. The parent directories, however, must exist before you issue the command. That is why, in the above two examples, you could not create both new directories with one command. The **cust** multidirectory had to exist before you could create the **s95** multidirectory in it.

Thank You

End of Session 10