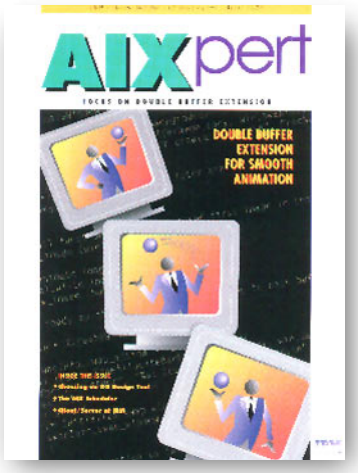
AIXpert DIGITAL VERSION IBM

# TABLE OF CONTENTS






APRIL
1996

AIXpert BACK TO MAIN MENU

# Gone Surfin'


Sometimes I sit down to write and the words just flow from my fingertips to the waiting keyboard. Each word, each syllable, falls smoothly into place like the gears of a well-lubricated, synchromesh transmission. This is not one of those times.

For almost three years I have been guiding *AIXpert* magazine, working with authors and communicating with our readers. We've put out a magazine each quarter packed with useful information to help programmers deal with the sometimes daunting task of writing applications for a very powerful and flexible operating system. We've helped with migrating to AIX 4, getting the most out of high availability systems, exploiting object technology, and optimizing databases, to name just a few of our topics. Putting together each issue has always been a challenge, but the rewards were great. From the warm reception that each issue received from our readers, we always knew that we were on the right track and the result was worth the effort.

But this issue is my last issue. For many months I have been doing double-duty as editor of this magazine, and also as editor of the online World Wide Web pages that support solution developers on all of IBM's platforms. I leave *AIXpert* to devote my full efforts to the Web pages. So, the next time you're out surfin' the Web, please come visit me at URL:

http://www.developer.ibm.com/

But before you do, tarry awhile over the pages of this issue of *AIXpert*. Learn about animation techniques using X-Windows' double buffering extensions, compare threads programming on AIX with OS/2 Warp, or better understand how to choose OO tools. Find out how to balance response-time critical applications with throughput-oriented programs running on the same system, browse through our Question and Answer section, or find your niche in one of the other interesting articles. You are sure to find something in this issue that is worth your time (and ours).

George Noren

---

**George Noren**, IBM Corporation, Internal Zip 4103, 11400 Burnet Road, Austin, TX 78758. Internet: geo@austin.ibm.com. Since joining IBM in September 1979, Mr. Noren has written manuals for System/34, System/36™, and AIX on both the RT® and RISC System/6000® platforms, and was a member of the InfoExplorer™ design team. He has also worked as system administrator for several AIX server machines and their clients, and is currently responsible for the Prototype Evaluation Labs in Austin. Mr. Noren studied engineering at Illinois Institute of Technology, holds a BA in English from the University of Minnesota and an MBA from St. Edwards University in Austin.

**George Noren**

# Double Buffer Extension

By Jeanne Sparlin and Michael A. Cohen

Double Buffer Extension (DBE) is an Application Programming Interface (API) that allows developers to write software that provides smooth animation. This article provides hints for using DBE in applications.

The Double Buffer Extension is an X Consortium X Server protocol extension and Application Programming Interface (API) that allows application developers to write software that provides smooth animation. The Double Buffer Extension became an X Consortium standard in May 1995. This article discusses Double Buffer Extension Protocol Version 1.0.

## Types of Buffers

A high-level understanding of graphics hardware is necessary to understand the importance of double buffering. In the simplest case, a graphics adapter contains a single color frame buffer that gets scanned to a monitor for viewing. Complex graphics adapters can have multiple sets of special purpose buffers, such as depth, stencil, alpha, and accumulation buffers, as well as multiple sets of color buffers. Special purpose buffers, which are not visible, are used in many rendering techniques. Some graphics adapters contain sets of color buffers that can be configured in several ways. For example, there might be 24 bitplanes of color buffer that can be used as 24 bits of a single color buffer or as two 8-bit color buffers—one that is visible and one that is hidden. (*Note:* When there is a single 24-bit frame buffer or two 8-bit frame buffers, 8 bits are left over.)

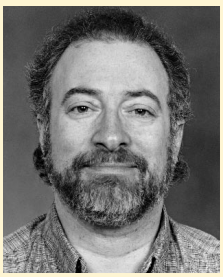
Multiple color buffers are used to produce smooth animation. An application draws to the back buffer while it is hidden. When the rendering is complete, the application swaps buffers. The frame buffer with the newly drawn picture becomes visible, and the formerly displayed buffer becomes invisible. The user instantly sees the new image without any artifacts of the actual rendering, such as tearing or the partial display of an image.

The technique of double buffering is analogous to a child's cartoon flip book that contains several pictures. In the flip book, the character's hand may move slightly in each frame to give the illusion that the character is waving. The same effect can be obtained by rendering each frame of the flip book in the back buffer, then swapping the back buffer to the front.

Buffer swapping was only available using the 3-D APIs in the initial releases of AIXwindows® and its associated 3-D APIs. The 3-D APIs in AIXwindows used a technique called *direct window access*. Rather than using the X Protocol to render, X Protocol reserves an area on the screen, then renders directly to the graphics adapter. Since the 3-D applications were a separate process from the X Server, the X Server was unaware that any buffer other than the one it was using was visible. This created a problem for applications that used the `GetImage` protocol to provide a screen capture facility. When the `GetImage` protocol request was received, the X Server grabbed the image from its buffer. So `GetImage` failed about 50% of the time because an alternate buffer was displayed.

In AIX 3.2.5, a fix for `GetImage` to the X Server combined with the MultiBuffer Extension (MBX) from the X Consortium provided a partial fix. The X Server now knew the currently displayed buffer and was able to retrieve `GetImage` data from the correct buffer. Since MBX was never an X Consortium standard, vendors implemented different prereleases of the extension. This resulted in confusion about the

Jeanne Sparlin



Michael A. Cohen

| Library Call | Function |
|---|---|
| XdbeQueryExtension | Returns the major and minor version numbers of the DBE extension |
| XdbeGetVisualInfo | Returns information about which visuals support DBE and their relative performance |
| XdbeFreeVisualInfo | Frees storage allocated by XdbeGetVisualInfo |
| XdbeAllocateBackBuffer | Allocates a drawable ID that refers to the back buffer of a window |
| XdbeSwapBuffers | Swaps the buffers for all windows listed, applying the appropriate swap action for each window |
| XdbeGetBackBufferAttributes | Returns information about a back buffer |

**Figure 1. Primary library calls**

semantics of some protocols and availability of some functionality.

MBX also uses an absolute method for displaying buffers. The application must explicitly draw in a specified buffer and keep track of what buffer is displayed. Absolute buffering did not meet the requirements of some 3-D APIs.

MBX was difficult to use and some supplied protocols, such as CreateStereoBuffers, were never implemented by any vendor. MBX was eventually discarded by the X Consortium and replaced by the Double Buffer Extension.

DBE uses relative buffering. When an application performs a swap buffer, the back buffer becomes the front buffer and the front buffer becomes the back buffer. The back buffer is always hidden and the front buffer is always displayed regardless of how many times a swap has occurred.

## The DBE Interface

The Double Buffer Extension to the X Window System® provides a standard interface to applications for buffer creation, deletion, and swapping. The Double Buffer Extension provides the following functionality[1]:

- Allocates and deallocates double buffering for a window
- Draws to and reads from the front and back buffers associated with a window
- Swaps the front and back buffers associated with a window
- Specifies swap actions to be taken when a window is swapped
- Requests that the front and back buffers associated with multiple double-buffered windows be swapped simultaneously

X client applications access DBE through the Xext extension interface library. Figure 1 lists the primary library calls, describes their function, and explains their use.

## Querying Extension Version

Extensions to the X protocol, like the X protocol itself, have major and minor version numbers. When protocol changes result in upwardly compatible software, the minor protocol number is increased. When protocol changes result in incompatible software, the major protocol number changes. Applications should query the major and minor protocol number when using X extensions to be sure that they can compatibly access the X Server portion of the extension.

An application should issue the XdbeQueryExtension before making any other DBE calls. This verifies that the application can compatibly access the X Server portion of the DBE extension and initializes the extensions.

If the application does not call XdbeQueryVersion, all subsequent DBE calls are undefined. This routine will return a non-zero value if the DBE library is compatible with the version returned by the X server. Zero is returned if the display does not support DBE or if there are other kinds of errors such as a communication failure with the server.

The sample program fragment in Figure 2 begins by making the XdbeQueryExtension call in order to start the use of double buffering: XdbeQueryExtension.

[1]Elliot, Ian (Hewlett-Packard) and Wiggins, David P. (X Consortium). *Double Buffer Extension Specification Protocol Version 1.0: X Consortium Standard.* 1995.

### Creating Windows Available for Double Buffering

Once the application knows that the DBE extension is available and the version is compatible, the application must create a window that can be double buffered. The application first determines the appropriate Visual, an X Window System resource that gives hints about the adapter hardware.

The `XdbeGetVisualInfo` subroutine returns a list structure that describes the buffering capabilities for the X Server. It can be called for one screen or for all screens in a multihead configuration. Each screen that the application wishes to query can be specified by a drawable, such as a window or pixmap. Typically, the application uses the root window of each screen as the drawable passed to `XdbeGetVisualInfo`. The code fragment in the next segment queries one screen. It uses the default root window of the display as the drawable that it passes to `XdbeGetVisualInfo`.

The following are returned for each visual that provides double buffering capability:

- Screen (useful in multihead situations)
- Depth of the visual
- Performance level of the visual

The visual with the highest performance level is likely to have better double-buffer graphics performance than the visual with the lower performance level. The returned performance level is only useful when comparing visuals for a specific graphics adapter. An application can make no assumptions about double-buffer graphics performance from the size of the difference of the performance levels or by comparing performance levels of various graphics adapters.

In AIX®, `XdbeGetVisualInfo` reports two performance levels: a high and a low value. The high value indicates that hardware buffers are available for this visual and a low value indicates that software buffers are available for this visual.

```
#include <X11/Xlib.h>
#include <X11/Xutil.h>
#include </X11/extensions/Xdbe.h>

...additional includes

/* Global variables */
Display  *dpy;

...additional global declarations

...function definitions

main (int argc, char *argv[]){

...declarations
int majorVersion
int minorVersion;
char *ProgName;

...parse command line args
...open display

/* see if we can use DBE */
if ( !XdbeQueryExtension (dpy, &majorVersion, &minorVersion) ) {
  fprintf (stderr, "%s: XdbeQueryExtension() failed.\n", ProgName);
  exit (8);
}
```

**Figure 2. Sample program fragment using DBE**

```
...declarations
Drawable              screen_list [1];
int                   num_screens = 1;
XdbeVisualInfo        *visInfo;

screen_list [0] = DefaultRootWindow (dpy);

if ( (visInfo = XdbeGetVisualInfo (dpy, screen_list, &num_screens) ) == NULL) {
  fprintf(stderr, "XdbeGetVisualInfo returned NULL\n");
   ...handle error condition
}

...application decides which visual to use

XdbeFreeVisualInfo (visInfo);

...application creates the windows that will be used for double buffering
```

**Figure 3. Sample program fragment**

Usually an application uses the highest performance visual that will support the requirements of the application.

After the visual IDs for visuals supporting double buffers are determined, they can now be used with other subroutine calls (for example, `XGetVisualInfo` and `XMatchVisualInfo`) that help an application decide what visual type is best. In addition to the ability to double buffer, when applications are selecting visuals, they usually consider the depth and the color class of the visuals. Color class for the X Window System includes grayscale, pseudocolor, direct color, and true color. The selected visual is then used in the `XCreateWindow` subroutine call.

Once the visual is selected, the `XdbeFreeVisualInfo` call is used to free storage returned by `XdbeGetVisualInfo`. In Figure 3, the sample program fragment makes the call to `XdbeGetVisualInfo`, selects the best visual to use based on application and performance requirements, then frees the storage associated with the query.

### Creating Double Buffers

To create a back (hidden) buffer for the window, the application calls the `XdbeAllocateBackBufferName` subroutine. When calling `XdbeAllocateBackBuffer`, the application uses the window that is created with the visual selected using the information from the `XdbeGetVisualInfo`. The window becomes the front or displayed buffer. The `XdbeAllocateBackBufferName` subroutine returns an X Window System resource called a back buffer (hidden buffer).

A back buffer is a new resource type defined by the Double Buffer Extension. In addition to window and pixmap, a back buffer becomes a drawable type. A back buffer can be used in any of the core X Window System rendering subroutines that accept a drawable parameter. Back buffers have the same depth and color class as well as the same clip list as their associated window. If the window is window shaped by using the X Non-Rectangular Window (Shape) Extension, the back buffer has the same shape as the associated window.

The application can begin drawing to this second or back buffer after a successful call to `XdbeAllocBackBufferName`. The sample program fragment is continued with the allocation of the back buffer and the building of data into it.

### Swapping Buffers

During normal processing, an application would draw a scene into the back buffer and then call `XdbeSwapBuffers` to make the back buffer visible. An application might even associate some type of timing loop to the draw/buffer swap actions if it desires a smooth animation.

`XdbeSwapBuffers` will swap front and back buffers according to the swap action for each window listed in the array of windows passed to

```
...declarations
XdbeSwapInfo swapInfo;
Window win;
XdbeBackBuffer buf;

win = XCreateWindow (dpy,.........);

swapInfo.swap_action = XdbeBackground;
if ( buf = XdbeAllocateBackBufferName (dpy, win,swapInfo.swap_action) == None) ) {
  fprintf (stderr, "%s: Couldn't create buffers\n", ProgName);
  exit(1);
}   else   {
  /* Initialize swap window for later use */
  swapInfo.swap_window = win;
}

/* Call function to initialize the first image in the displayed window */
draw_image (win);

...ready to start animation
```

**Figure 4. Allocating the back buffer and drawing initial image**

`XdbeSwapBuffers`. Note that each window swapped in a single call to `XdbeSwapBuffers` can have a different `swap_action`. The `XdbeSwapInfo` structure contains two fields:

- Window for which to swap buffers
- Swap action to be applied during that swap

A swap action specifies what the X Server should do with the front buffer when the buffers are swapped. The four types of swap actions defined by DBE (illustrated in Figures 5, 6, 7, and 8) are described below.

**Undefined.** The contents of the new back buffer become undefined. This may be the most efficient action since it allows the implementation to discard the contents of the buffer if necessary. The application is not guaranteed anything about the state of the back buffer after the swap.

An application might use this swap action if it desires fast swaps and does not care what happens to the back buffer. For example, a specialized application might always copy an existing image from off-screen storage into the back buffer before it begins rendering. This could be a landscape or some other static background.

The application does not care what happens to the new back buffer when it is swapped. It is going to overwrite the entire buffer anyway.

**Background.** The unobscured region of the new back buffer will be tiled with the window background. The background action allows devices to use a fast, clear capability during a swap. This action might be used if the application is not rendering the entire buffer and wants the rest of the buffer filled with the window background. In one atomic protocol the buffer is swapped and the new back buffer is cleared and made ready for the next rendering. The application saves time because it does not have to send another protocol to clear the back buffer before it can begin rendering.

**Untouched.** The unobscured region of the new back buffer will be unmodified by the swap. This swap action might be used if an application is only going to modify a portion of the scene. It does not have to erase or rerender part of the scene. It just enhances what is already there. For example, an animation of a

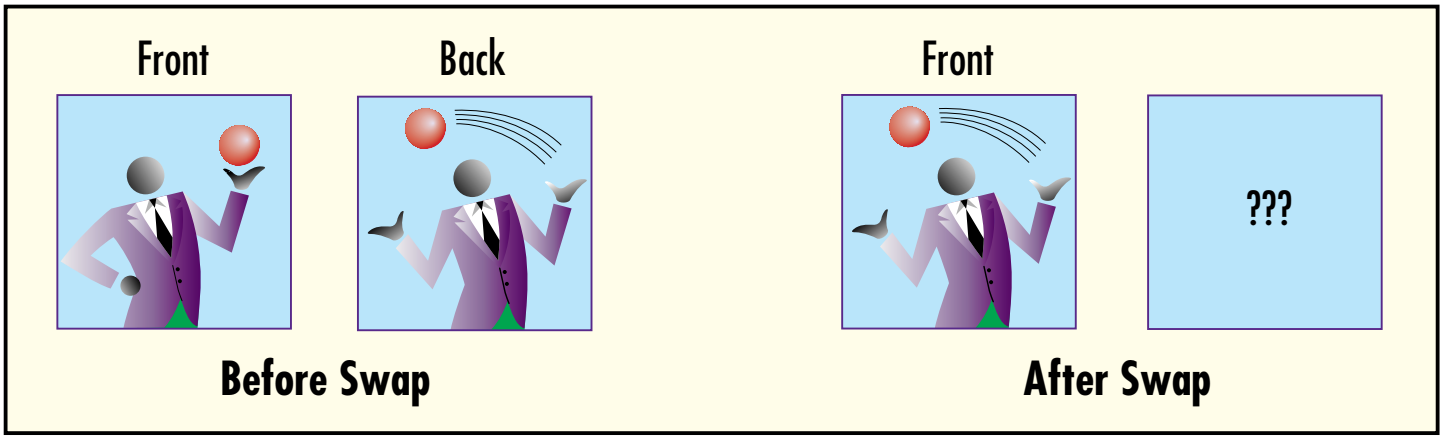



Figure 5. Undefined

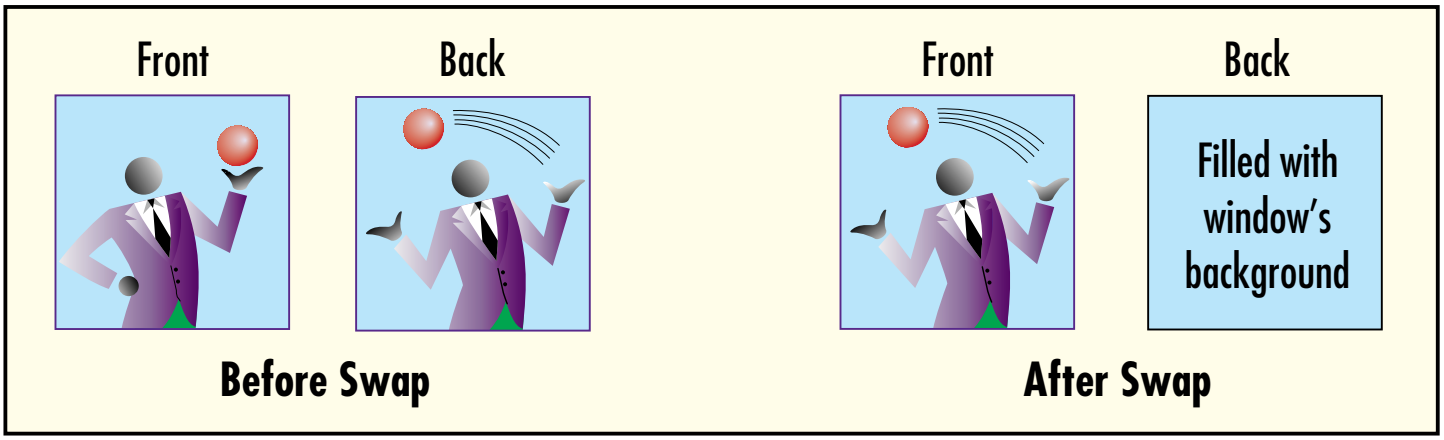



Figure 6. Background

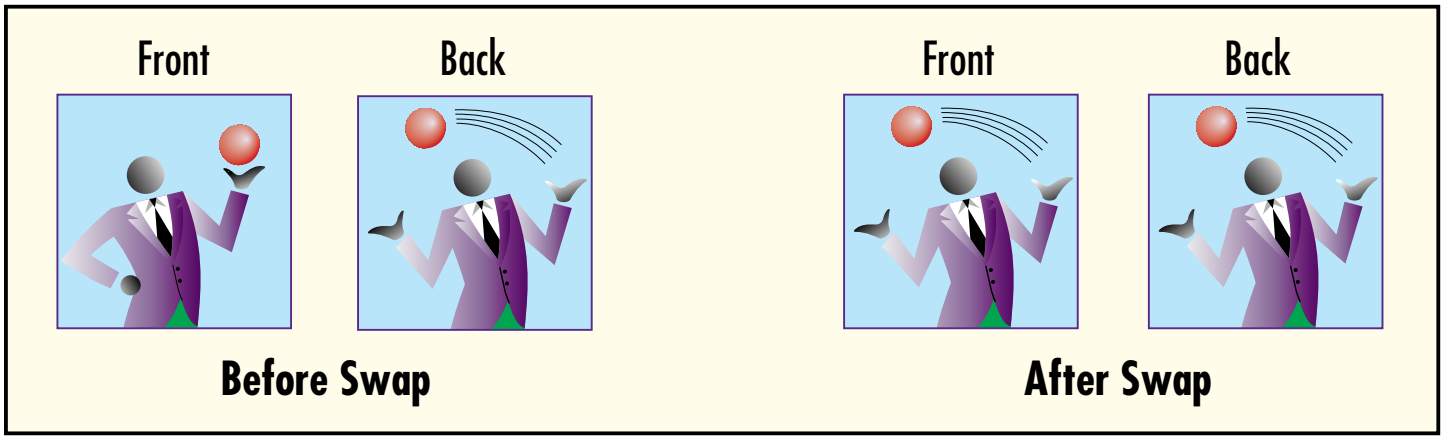



Figure 7. Untouched

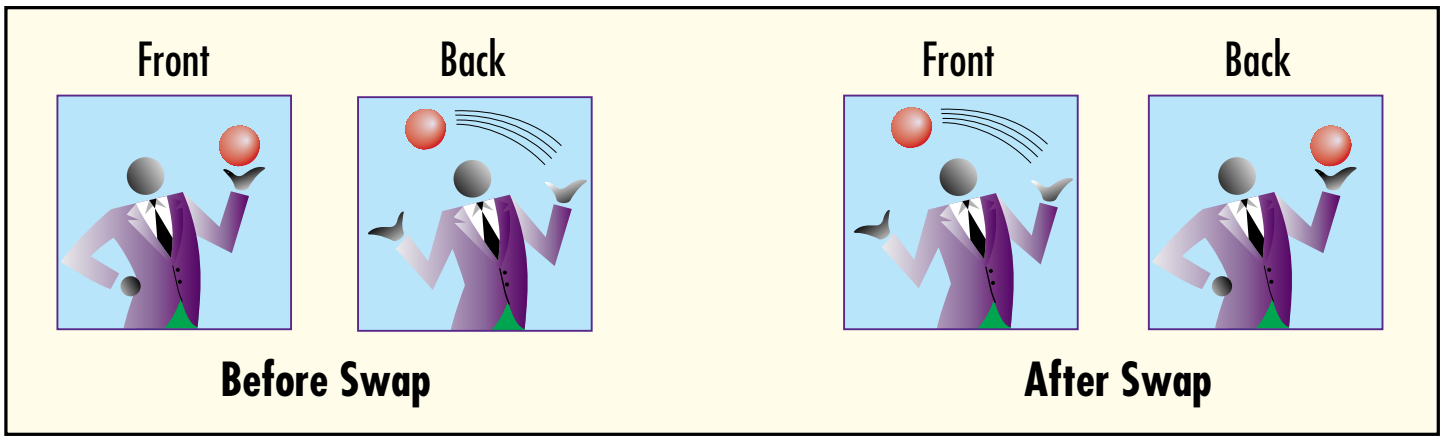



Figure 8. Copied

```
/* loop until some action ends the animation */
for (;;) {
  /* animate image by moving it slightly and drawing
     new image to the back buffer */
  recalculate_image ();
  draw_image (buf);
  XdbeSwapBuffers (dpy, &swapinfo, 1);
  if (we_are_done_animating())
    break;
}
```

**Figure 9. Animating using XdbeSwapBuffer**

house being built would add something new to each frame and not rerender the entire frame.

**Copied.** This swap action might be used if an application knows that it must maintain a high degree of data integrity. The newly exposed region will contain inconsistent data for any swap action besides Copied when a window is exposed. When the swap action is Undefined, there might be garbage; when the swap action is Background, the region will contain only a color or pixmap. Until the application can redraw the back buffer and swap, the newly exposed regions may look inconsistent. When the swap action is copied, the newly exposed regions remain consistent with the rest of the scene.

Figure 9 continues the sample program fragment with the use of `XdbeSwapBuffers`.

## Conclusion

The Double Buffer Extension is a simple interface for software developers to incorporate smooth animation into their application programs. DBE provides an X Consortium standard API that was not present in previous versions of AIXwindows. DBE is available for both 2-D and 3-D developers. The Double Buffer Extension ships with AIX 4.1.4.

**Jeanne Sparlin**, IBM Corporation, 11400 Burnet Road, Austin, TX 78758. Ms. Sparlin joined IBM in 1983 after completing her computer science training at the University of Texas, Austin. She has previously worked in the Data Management, SNA, Dialog Design Aid, and the X Window System development departments on the RT PC®. She joined the Graphics Architecture department in October 1988 and currently does X Window System architecture. She also represents IBM on the Advisory Committee of the X Window System Consortium. She has a BS in Education from Northeast Missouri State University.

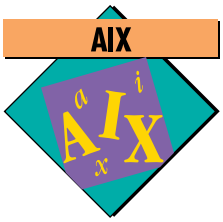
**Michael A. Cohen**, IBM Corporation, 11400 Burnet Road, Austin, TX 78758. Mr. Cohen joined IBM in 1988 and has participated in a variety of projects for the IBM graphics organization. He was a ddx programmer for the X Server for the GXT500 graphics adapter and, more recently, the lead developer for the Double Buffer Extension on several IBM RISC System/6000® platforms. He has a BS from Purdue University and a Master of Science in Computer Science from the New York Institute of Technology.

## IBM Launches Virtual Pavilion

IBM has launched its "Small Planet Pavilion" Web site as part of the Internet's 1996 World Exposition. Expo '96 is a year-long event that will take place in cyberspace and in "real space" in schools, museums, and cultural centers around the world. Events will be held throughout the year on the Expo's "electronic fairground."

The IBM Small Planet Pavilion lets people experience the power of a networked world. The pavilion will be a participatory environment where individuals "learn by doing." The IBM Small Planet Pavilion can be found on the World Wide Web at http://www.ibm.park.org.

# Process Priorities and the AIX Scheduler




By Allan Pellnat and Donald Totten

This article provides application developers and system designers insights into using process prioritization to achieve user satisfaction when both response time-oriented and throughput-oriented applications must run together.

The AIX process priority schema and scheduler are effective in delivering a fair share of the CPU resource to each user. Mixing response time-oriented transaction applications with throughput-oriented batch operations on the same system is the classic case for process prioritization. Fine-tuning key application process priorities makes the AIX Scheduler's performance even better and keeps end users happy. To do this, developers need detailed knowledge of how processes fit together to provide an end-user application.

## The Environment

Directory One™ is the automated Directory Assistance (DA) service of Nortel™ Directory & Operator Services division of Northern Telecom®. It has established feature and performance benchmarks for the telephone industry worldwide. Telephone companies require their automated DA systems to ensure rapid response time for operator searches of very large databases under heavy system loads every day around the clock.

Directory One currently runs on IBM RISC System/6000 (RS/6000™) processors using AIX Version 3.2.5. Plans are underway to migrate to AIX Version 4.1 and Symmetric Multiprocessing (SMP) processors. The Oracle® database is used for the listings.

## The Problem

Databases of telephone numbers contain millions of listings. Because as much as 1% of these databases are updated daily, the processes for updating and backing up the databases must ensure transaction throughput without degrading DA operator search response time.

The challenge is to deliver rapid search response time during the CPU intensive update and backup periods without significantly reducing the update and backup throughput. The quick and easy answer seemed to be boosting the process priority of key search application processes above the update and backup application priorities. But that proved to be wrong.

## The Approach

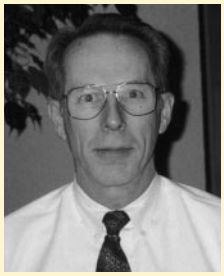
The test consisted of RS/6000 processors handling simulated DA calls and searches, while a batch update process added and deleted listings to the database. Search response time was measured by trapping and time stamping search requests and responses with a protocol analyzer on the Fiber-optic Data Distribution Interface (FDDI) ring. Update throughput was calculated from log file start and end times, and the known number of add and delete transactions.

Figure 1 shows the process prioritization test bed.

### Response Time Measurements

We needed a data presentation method that would clearly show us the effect of changing process priorities. We chose a graph of the cumulative distribution function for the probability of search response time. Plotting the results of each test series against a known baseline makes the relative differences readily apparent.

Baseline measurements of search response time were taken for a "no load" case consisting of 600 different searches allowing the AIX Scheduler to "nice" the process priority values up from the starting value of 60. (*Note:* A process whose priority value has been raised above the 60 starting point has lower precedence in gaining access to the CPU.)



*Allan Pellnat*

*Donald Totten*

Trace A in Figure 2 shows these results. The graph shows the probability of search response time being less than a given value. Under no load conditions—those with minimal queuing—the average response time will occur approximately at P = .67. The average response time for this baseline test was 40 ms.

Figure 2 shows search response time probabilities for three cases with AIX doing dynamic prioritization of processes. Trace A is the no load baseline condition where search response averages 40 ms. Trace B shows that a search-and-call load that utilizes 60% of the CPU results in an average search response of 289 ms under the AIX dynamic prioritization regime.

The critical case is Trace C, the 60% call-and-search load concurrent with a batch database update application. Average search response time increased to greater than one full second, which was totally unacceptable.

## Roles of the Processes

The batch update process added 1,800 listings, then deleted the same 1,800 listings. Under no load conditions, the batch update completed the 3,600 transactions in 38 minutes. Concurrent with the 60% CPU load of search-and-call control, batch update did the 3,600 transactions in 61 minutes. This is an acceptable throughput rate of one transaction per second.

The solution to the problem lies in understanding the role that each of the six separate processes plays in delivering the whole call control and search application.

In some cases, a single instance of a process handles all transactions for all calls or searches. In other cases there may be multiple instances of a process, in which each instance handles one or more calls or searches concurrently. Although a typical call lasts for less than 30 seconds in normal operations, hundreds of calls can be active in various states at any point in time. On average, each call generates two searches. Two servers sequentially handle searches of a particular database. A single, dedicated process per communication link handles all communication between the outside world and the call control processes.

Earlier work had established the per-call and per-search average service times for each of the six unique processes that together form the primary application of the system. The distribution of work per call among the six processes is characterized in Figure 3.

In a typical business day, peak call-and-search loads that occur in mid-morning and mid-afternoon are approximately equal. Total CPU utilization of the RS/6000 processors may exceed 70% for an hour or more at these peak times. The six critical search-and-call control processes account for the lion's share of this CPU utilization.

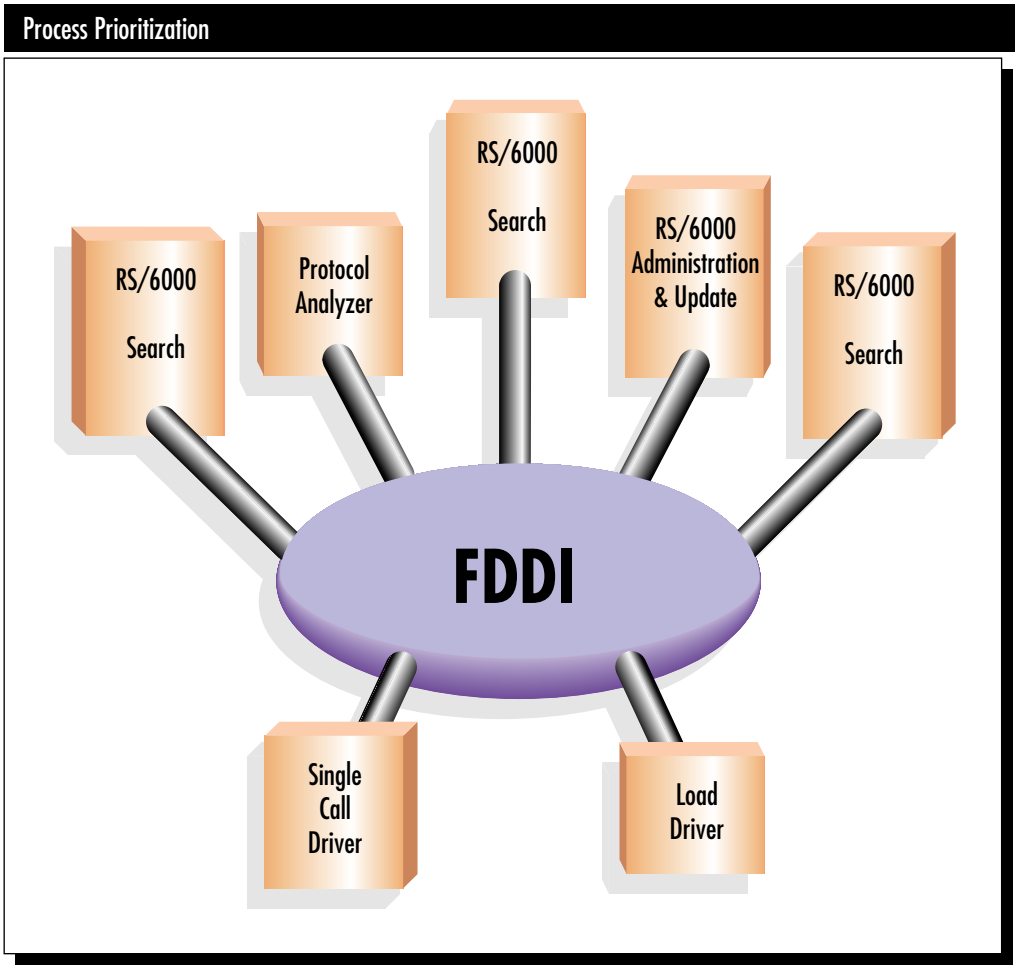



**Figure 1. Process prioritization test bed**

Moving all six processes to a higher fixed priority creates a problem: Process B, a single process that consumes the largest amount of CPU time on a per call basis, starves out the other five processes. Every time a CPU context switch takes place, there is a high probability that Process B will be next in the run queue because it is used so many times during a call.

It was evident that simply prioritizing the six key processes over the other workloads on the system would probably cause more harm than good. The solution to the problem was to understand how the six separate processes work together to deliver the application.

Process A—the search—accounts for 48% of the per call CPU demand. Because any given search is served by one of several instances of Process A, the average utilization of any of those processes is only a fraction of the total CPU time for a call. Similarly, Process C can have ten or more instances, which results in a small amount of actual usage on a per call basis for any single instance.

However, Process B is only a single instance called to perform some processing seven or more times during every call. The AIX Scheduler dynamically prioritizes these processes based on short-term CPU utilization of each process. Process B, the "hog" of the family, gets "niced" out of the way by the AIX Scheduler. When its priority is fixed, it tends to starve out the other processes, especially the key element, Process A.

## The Solution

The solution is to set each process to a fixed priority based on its contribution to the key measured element, the response of Process A—database search. Figure 4 shows the processes again with their percentages and fixed priority settings.

The search process itself gets top priority. Search requests and responses pass through Processes B and C, so they are next in order of preference. The remaining processes are not directly in the search path. Getting them into and out of the CPU quickly assures that they are not holding up the critical search processes. They are prioritized equally at a level above other jobs running in the system, but below the critical search-related processes.

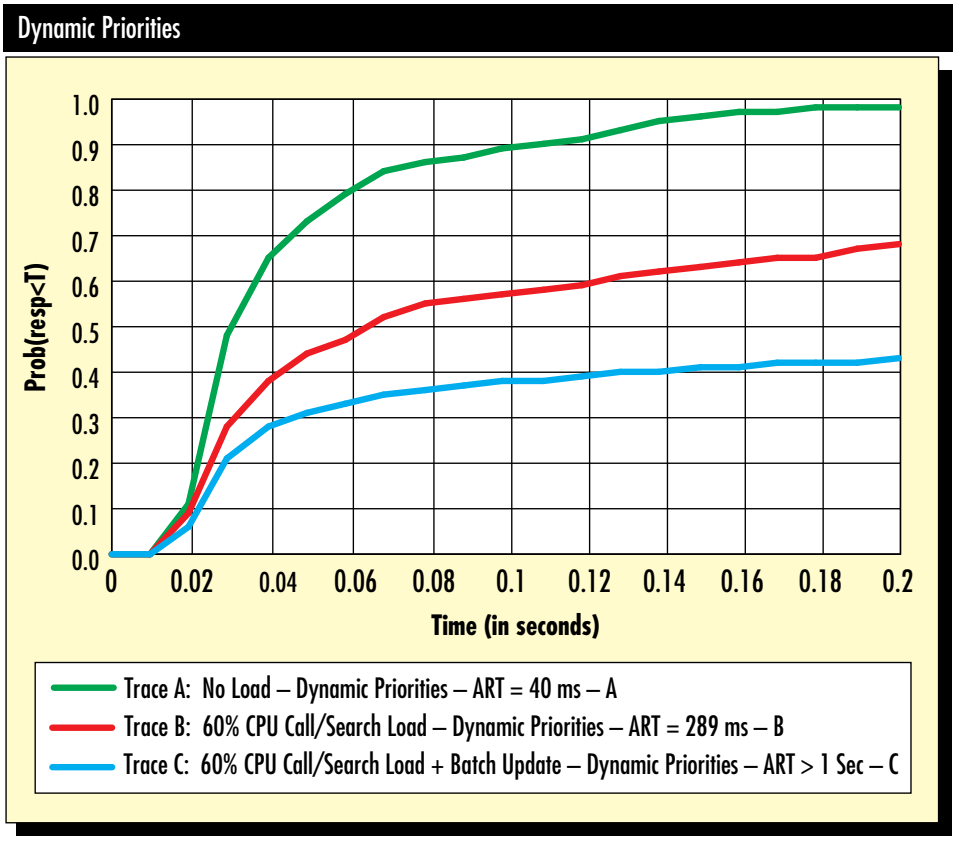



**Figure 2. Search response time: no load and 60% CPU load + batch update = dynamic priorities**

| Process | Per Call Work Proportion | Work Distribution |
|---|---|---|
| A | 48% | Runs twice during a call |
| B | 26% | Runs seven or more times during a call |
| C | 16% | Runs four or more times during a call |
| D | 4% | Runs four or more times during a call |
| E | 4% | Runs once at the very end of a call |
| F | 2% | Runs one to three times during a call |

**Figure 3. Work distribution per call**

| Process | Per Call Work Proportion | Work Distribution |
|---|---|---|
| A | 48% | Fixed Priority 45 |
| B | 26% | Fixed Priority 46 |
| C | 16% | Fixed Priority 47 |
| D | 4% | Fixed Priority 48 |
| E | 4% | Fixed Priority 48 |
| F | 2% | Fixed Priority 48 |

**Figure 4. Fixed priority settings per call**

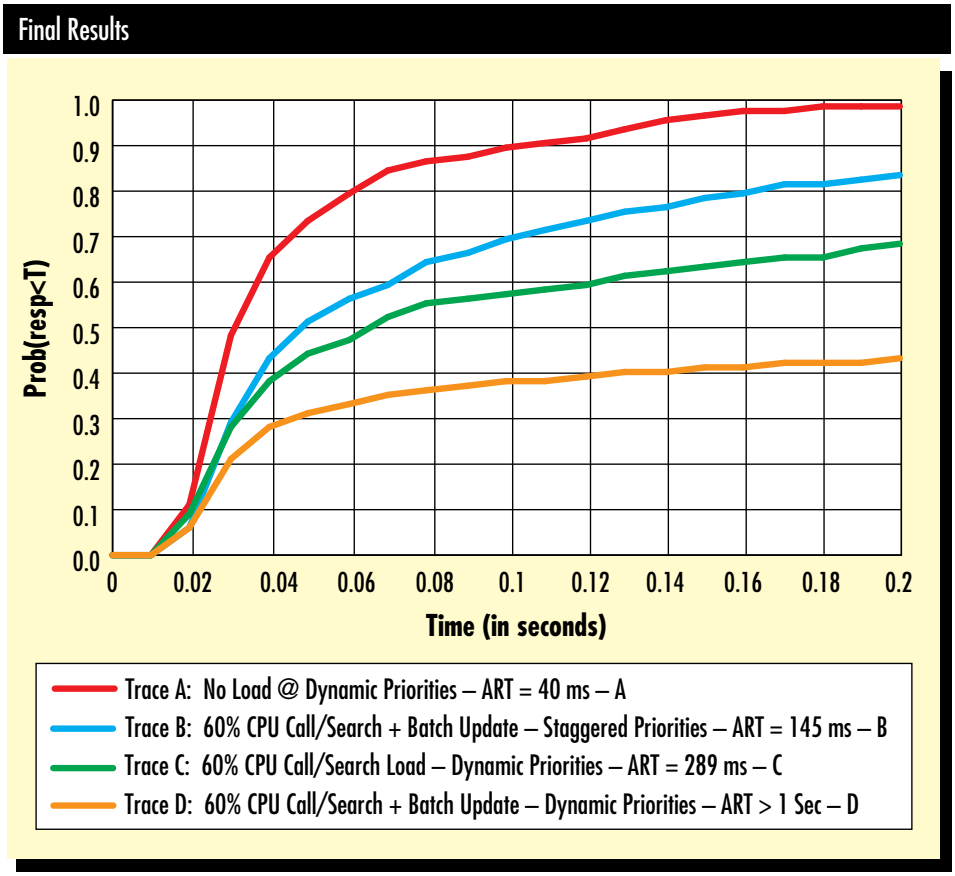



**Figure 5. The 60% CPU load and batch update—staggered fixed priorities**

Figure 5 illustrates the dramatic effect of this scheme on search response time probability. Together, the full 60% call-and-search load and the batch update process now yield an average search response time of 145 ms—twice as good as the call-and-search load alone under dynamic priorities.

Although this slows down the batch update throughput, it is still within the acceptable range. Under dynamic priorities, the 3,600 batch transactions required nearly an hour to complete. Under the staggered fixed priorities, the same operations took 70 minutes—a 16.7% change. This represents a worst-case scenario, however, for the concurrent batch update. Under normal operating conditions, batch database updates are run in the early hours of the morning when the call-and-search volume consumes considerably less than 60% of the CPU.

## Conclusion

Identifying a key application to receive priority access to the shared CPU resource can backfire unless you understand the inner workings of that application. Applying process prioritization to a block of application processes without analyzing the interaction of the processes may lead to poorer performance rather than better. AIX does a very good job of scheduling fair-share CPU usage among diverse processes and applications. When application developers and system designers cooperate in using AIX performance tuning tools, end users reap the benefits.

**Allan Pellnat**, Northern Telecom, Directory and Operator Services, 97 Humboldt Street, Rochester, NY 14609. Internet: agp@sun111.cci.com. Mr. Pellnat is an advisory engineer specializing in defining and evaluating overall system performance for the Directory One product.

**Donald Totten**, Northern Telecom, Directory and Operator Services, 97 Humboldt Street, Rochester, NY 14609. Internet: djt@nt.com. Mr. Totten is an advisory engineer responsible for specification, validation, and verification of product performance. He has a BS in Mathematics and a BS in Computer Science from SUNY Brockport.

# Thread Programming Models: AIX 4.1 vs. OS/2 Warp

By Chary G. Tamirisa

This article presents a comparison of the multithreading models supported in the AIX and OS/2® operating systems. It focuses on the key differences in the two programming models, without describing the details of the Application Programming Interfaces (APIs). The goal is to assist porting of applications between the two platforms. This article assumes the reader is familiar with one of the two models.

The AIX 4.1 and OS/2 (32-bit) operating systems offer powerful programming models for multithreaded programming. The AIX model is based on the IEEE's® POSIX.1c Draft 7 standard for threads whereas the OS/2 thread model is an IBM-specific model. Increasingly, software is being ported from one platform to the other, which creates a need to understand the execution models offered by both systems to simplify application porting. The comparative study of the two threading models that follows does not include details on the actual programming interfaces.

Familiarity with either the AIX 4.1 or the OS/2 model is essential in order to understand this article. The conceptual foundations for the thread services for OS/2 can be found in the OS/2 Warp™ Control Program. The AIX online publications provide extensive documentation of the thread support.

The threads programming model deals with thread creation, mechanisms for synchronization of threads, thread-local memory, thread cancellation, and graceful handling of asynchronous events in a multithreaded process.

## Overview

The AIX[1] thread model is based on the POSIX.1c Draft 7 standard for threads. The supported interfaces are a subset of the interfaces defined in the POSIX.1c standard. POSIX.1c specifies the following optional features:

- Stack address for a thread to be specified by the user
- Support for interprocess-shared mutexes and condition variables
- Support for process-scope contention for threads instead of all threads competing at system scope
- Support for protocols that prevent priority inversion in which a low-priority thread runs while a high-priority thread is blocked

AIX does not support these optional features.

The OS/2 thread model shares several common features with AIX. It implements system-scope for threads and allows the thread stack size, but not the stack address, to be specified. It does not provide support for preventing priority inversion. However, OS/2 supports an important feature called *interprocess-shared synchronization*, which allows threads from different processes to synchronize with one another using the mutex and event semaphores.

AIX supports extension of UNIX® signals into a multithreaded process; OS/2 maps signals to exceptions. AIX and OS/2 support process-wide timers. AIX generates timer signals whereas OS/2 uses the semaphore events to inform the process of timer events.

Chary G. Tamirisa

[1] All references to AIX throughout the article refer to AIX 4.1.

The notable differences appear in the synchronization services between AIX and OS/2. The AIX condition variable mechanism differs completely from the OS/2 event semaphores. OS/2 allows mechanisms in which a thread can stop one or all other threads from being scheduled. Since these types of features are very powerful, they should be used very carefully. AIX differs because it has no provision for one thread to directly stop other threads.

The sections that follow provide a detailed comparison of the AIX and OS/2 threading models.

## Mapping User and Kernel Threads

AIX and OS/2 map one user thread to one kernel thread called 1:1 mapping. All application threads create corresponding kernel threads. Since each kernel thread uses important kernel resources, the operating system limits the number of threads that can be created in the user process. For example, in AIX each process can have up to 512 threads. In OS/2, each process can have up to a specified (in `CONFIG.SYS`, it is typically 256) number of threads, but subject to a system-wide maximum of 4095 threads.

**Threads are the scheduling units of execution in AIX and OS/2.**

Threads are the scheduling units of execution in AIX and OS/2. Since all threads are scheduled on a system-wide basis, there is no process-scope scheduling. Due to the system-wide scheduling of threads in both operating systems, applications that create high-priority threads to boost the application performance can adversely affect scheduling of threads from other processes.

## Thread Creation

The thread creation Application Programming Interfaces (APIs) for OS/2 and AIX allow the thread function to be specified and provide for an identifier of the created thread to be returned. However, AIX provides a thread attribute parameter that allows threads to be created with the following properties that can be set:

- Thread detach state
- Thread stack size
- Thread stack address
- Thread scheduling inheritance property
- Thread scheduling policy and scheduling priority
- Thread contention scope

OS/2 has no explicit thread attribute parameter when a thread is created. OS/2 allows a flag to be specified that shows whether the newly created thread is ready to run or in a suspended mode. In addition, the flag can also specify whether the memory for the stack is sparse or committed.

## Detach/Join

AIX allows two types of threads:

- **Joinable threads:** A thread that allows other threads to join with the newly created thread. A joinable thread must be either joined or detached explicitly in order to free the state (memory) associated with it. Otherwise, memory will be associated with these threads and not released when they are terminated, which creates memory waste.

  OS/2 threads are always joinable. If a thread joins with an already terminated thread, an error code is returned as the state associated with an already terminated thread is freed. Porting the AIX joinable threads to OS/2 requires explicit synchronization through the OS/2 event semaphore APIs.

- **Detached threads:** A thread in which the state associated with it is freed automatically when the thread exits. AIX allows threads to be created in the detached state (this is also the default state). Such threads cannot be joined by other threads. OS/2 detaches threads automatically on their exit.

## Stack Size/Address

AIX and OS/2 enable an application to set the stack size. Although POSIX.1c allows the thread stack address to be specified, AIX does not currently support specification of the stack address. OS/2 does not allow specification of the stack address for a thread. AIX does not allow applications to control the management of the thread's stack. However, OS/2 allows committed stack memory and sparse memory to be created.

## Scheduling Inheritance

In AIX, the default thread creation attribute results in the newly created thread inheriting the scheduling policy and scheduling priority from the creator thread. Similarly, an OS/2 thread inherits the scheduling class and level of the creating thread.

Setting the scheduling class and priority level of a thread is possible in both AIX and OS/2. One difference, however, between AIX and OS/2 is that AIX allows the thread scheduling policy and priority to be determined by specifying the thread attribute parameter when a thread is created. This makes it possible to create a thread with a different scheduling policy and priority than the creating (parent) thread. When porting AIX applications to OS/2, if the AIX application creates threads with scheduling attributes that differ from those of the creating thread, then the process is different in OS/2. To create the same functionality in OS/2, use the suspended mode for the new thread, then use the OS/2 priority setting API to modify the scheduling class and level of the suspended thread and then resume it.

## Thread Termination

A thread can terminate itself in one of two ways: by returning from the thread function or by calling an explicit API to exit the thread. AIX and OS/2 support both types of thread termination. In AIX, `pthread_exit()` in the main thread does not terminate the process; it only terminates the main thread. In OS/2, exiting the main thread in any way terminates the process.

## Exit Status

AIX allows a thread to pass its exit status information to another thread through the `pthread_join()` API. In addition, AIX allows cleanup handlers for thread-local memory to be registered. OS/2 does not provide support for a thread to inform another thread in the same process of its exit status. The `DosExit()` API allows exit status of a child process to be available for the parent process.

## Thread Suspension

AIX has no API to suspend another thread; each application must explicitly create synchronization so that threads can be suspended or resumed.

OS/2 allows a thread to suspend another thread through `DosSuspendThread(threadid)`. In addition, the OS/2 API `DosResumeThread()` allows a thread to resume the suspended thread.

### Wait for Thread to Terminate

In AIX, the `pthread_join()` API allows a thread to wait for termination of only one other thread; there is no direct API to wait for multiple threads as in OS/2. This call blocks if the target thread did not terminate.

AIX requires the use of explicit synchronization APIs to achieve the OS/2 functionality of waiting for termination of multiple threads. In OS/2, the `DosWaitThread()` API allows a thread to wait for any thread—or a specific thread—in the process to terminate. It also allows a thread to determine if another thread has terminated.

## Thread Information

In AIX, a thread identifier can be obtained for the current thread by a call to `pthread_self()`. It is often convenient to refer to a thread using a small integer value rather than an address. For this purpose, the extension `pthread_get_unique_np()` is available (note POSIX™ allows such non-portable extensions with a suffix of `_np`). This is helpful when indexing an array with a thread identifier. However, since the IDs can be recycled, it is important to clean up the data after a thread exits.

OS/2 allows a thread to obtain thread-specific information by a call to `DosGetInfoBlocks()`. This API provides the current thread ID—a small integer value.

In AIX, the thread stack size and stack address can be obtained by calling the `pthread_attr_getstacksize()` and `pthread_attr_getstackaddr()` APIs respectively. Similarly, the scheduling policy and priority can be obtained dynamically by calling `pthread_getschedparam()`.

In OS/2, a call to `DosGetInfoBlocks()` provides both the thread and the Process Information Blocks. The available information includes the stack pointer, stack limit, thread priority class, and thread priority level.

## Synchronization

AIX supports mutual exclusion and thread synchronization through the condition variables. This support is limited to threads within the same process, and currently does not support mutual exclusion between threads in different processes, or condition variable synchronization between threads in different processes.

OS/2 provides three types of semaphores for thread synchronization within the same process or with threads across processes. The three types of semaphores include Event, Mutex, and Multiple Wait semaphores (Mux semaphore). Multiple Wait semaphores allow a thread to wait for several event or mutex semaphores. AIX does not support the Multiple Wait semaphores; however, they can be supported using the mutexes and condition variable APIs.

**A thread can terminate itself by returning from the thread function or by calling an explicit API to exit the thread.**

```
pthread_mutex_t mutex; /* Lock the associated mutex */
pthread_cond_t condition;/* Condition Variable */
int flag;/* Boolean */

..

pthread_mutex_lock(&mutex);

while(!flag)
pthread_cond_wait(&mutex, &condition);
/* Release the mutex and block atomically */

pthread_mutex_unlock(&mutex);
```

**Figure 1. AIX condition variable usage**

### Mutexes

AIX and OS/2 APIs that allow threads to guarantee mutual exclusion are similar. AIX supports three types of mutexes: recursive, non-recursive, and fast (non-recursive with no error checking). These mutexes can be created by specifying mutex attributes when initializing a mutex. AIX 4.1.4 does not support mutexes between threads in different processes.

The OS/2 mutex semaphores are similar to the AIX recursive mutexes. OS/2 also supports mutex semaphores to allow mutual exclusion between threads in different processes. Therefore, porting applications to AIX that depend on process-shared mutexes requires some porting effort. A hint is to use the AIX atomic instructions (`_check_lock()`) and to use shared memory to obtain process-shared mutex functionality. Another approach may be to write a kernel extension that provides the process-shared mutexes on AIX.

### Condition Variables

AIX supports condition variables to allow threads in the same process to synchronize among themselves. Associated with the AIX condition variable are a boolean predicate and a mutex. Together these ensure an atomic operation in which a thread acquires a mutex lock and checks to see if it should block. If it should block, it blocks and releases the lock atomically. The code segment in Figure 1 captures the idiom, illustrating the case of condition variables. The condition variable mechanism differs from the OS/2 event semaphores, which are similar to counting semaphores (a counting semaphore is one that remembers the number of times an event is posted). The significant difference between an OS/2 event semaphore and a counting semaphore is in the wake up when an event is posted. The OS/2 event wakes up all waiting threads whereas a counting semaphore wakes up one of the waiting threads.

The OS/2 event semaphores allow the post count to be set to zero automatically after the query; therefore events that are posted are not lost.

Since semaphore waiting can be indefinite, a timeout parameter is often very useful to keep application threads from blocking forever. Both AIX and OS/2 allow a timeout parameter while waiting for an event.

AIX supports interfaces that wake up either one waiting thread or all the waiting threads. OS/2 event semaphore posting provides a broadcast semantic—all waiting threads for this event are awakened.

The event semaphores in OS/2 allow an interrupt (exception) handler to post on an event semaphore. The AIX condition variables mechanism with its associated mutex is not generally safe to be invoked from signal handlers because self-deadlock may occur.

### Thread Scheduling

AIX provides two types of scheduling priorities: fixed and variable. The real-time scheduling policies—First-In-First-Out (FIFO) and Round-Robin (RR)—are in the fixed priority class where the operating system will not change the priorities selected by the application.

AIX supports a UNIX type of variable priority scheduling policy called `SCHED_OTHER` under the second category.

In OS/2, thread scheduling behavior is determined by the priority class and the priority level to which it belongs. The priority class and level are similar to the AIX thread scheduling policy and scheduling priority respectively.

OS/2 provides four scheduling classes (policies) that allow applications to set threads with scheduling behavior that is appropriate to the job they do:

- **Time-Critical.** Class of threads with the highest priority; since they run with fixed priorities, the OS will not adjust the priorities of these threads
- **Fixed-High.** Scheduling class (policy) that allows the OS to adjust its priorities under the `DYNAMIC` option; threads have higher scheduling priority than those with the Regular class
- **Regular.** Scheduling class (policy) that allows the OS to adjust its priorities under the `DYNAMIC` option

- **Idle-Time.** The last class in scheduling class priority; has the lowest scheduling priority; when no other work is to be done, threads with this priority class get to run

OS/2 provides for round-robin scheduling of threads with equal priority within the same class. The OS/2 `TIME-CRITICAL` statement corresponds to the round-robin scheduling policy in AIX. Under the `DYNAMIC` option, the Fixed-High and Regular scheduling classes are similar to the `SCHED_OTHER`.

AIX offers 127 priorities (1–127); all scheduling policies share this range. This means there can be multiple threads within a process that have one priority value, but have different scheduling policies. Although threads are scheduled based on their priorities, their scheduling policies determine what happens when a thread gets preempted. OS/2 scheduling is based on a similar model. OS/2 offers 32 priority-level values (0–31) within each class. These scheduling classes govern thread scheduling.

In AIX, an application needs the effective user ID of the root to create threads with fixed scheduling policies. This is not necessary in OS/2.

It is not possible in AIX to specify scheduling behavior for all threads on a system-wide basis. In OS/2, two types of scheduling behaviors can be chosen system-wide. If the `PRIORITY` statement in `CONFIG.SYS` is set to `ABSOLUTE`, the thread priorities are fixed by the applications. In this case, the operating system will not change priorities of threads. However, if the `PRIORITY` is set to `DYNAMIC`, the priorities of threads are adjusted based on system load and process activity, and on whether the process is doing interactive I/O.

There are two exceptions to the dynamic adjustment that occurs in the Time-Critical and Idle-Time thread classes. The default value for `PRIORITY` is `DYNAMIC` and this allows the operating system to achieve optimal performance. If this is not sufficient, specifying `ABSOLUTE` option for `PRIORITY` enables applications to have better control of the scheduling behavior. AIX has no such system-wide modification of thread scheduling policy or priority. The system implements certain documented scheduling policies and an application can create a thread based on these policies. In AIX, threads created with the default thread attributes share the same scheduling policy and priority.

### Scheduling Time Slice

In AIX, an application cannot choose the thread scheduling time slice. In OS/2, the minimum and maximum value that controls the time allocated for threads can be specified by using the `TIMESLICE` statement (in `CONFIG.SYS`). The min value must be an integer greater than or equal to 32 milliseconds. The min value specifies the minimum time that a thread is processed before yielding CPU to another thread of the same priority. The max value must be an integer within the interval of 32 to 65536 milliseconds. The max value specifies the maximum amount of time a thread can be processed before yielding CPU.

### Inheritance of Priority

In AIX, when default thread creation attributes are used, an application can create a thread with a different scheduling policy/priority than its own. In OS/2, the priority class and level of the newly created thread are set to the same as the thread that created it.

AIX does not allow a thread to change the priority of another running thread or to change the priority of threads in another process. OS/2 allows the thread to change the priority of threads within the process or the priority of those threads in processes that are its children.

**AIX does not allow a thread to change the priority of another running thread or to change the priority of threads in another process.**

## Once-only Semantics and Critical Sections

For applications and libraries to ensure that shared resources such as mutexes and condition variables are initialized once-only in a multi-threaded environment, AIX provides the API `pthread_once()`. This API guarantees that only one thread invokes the specified once-only initialization function.

OS/2 allows a thread to stop any other thread in the same process from being scheduled when an application invokes the `DosEnterCriticalSection()` API. This can lead to dead-locked threads that may hold resources needed by the thread that invoked `DosEnterCriticalSection`. To avoid this, applications must ensure that this API is invoked when no other thread can hold resources needed by the calling thread. This API is typically invoked, for example, to ensure that initialization of semaphores is done only once in a Dynamic Link Library (DLL).

## Thread-Specific Data

AIX provides the interfaces to create, destroy, get, and set thread-specific data. OS/2 provides

interfaces to allocate thread-specific data and to free the data. In AIX, approximately 1,000 thread-specific keys can be allocated, whereas OS/2 allows a maximum of 128 bytes to a thread for associating thread-specific data.

In AIX, the thread-specific data can be destroyed automatically by associating a destructor upon the data creation. Since no destructors are associated with thread-specific data, the application must explicitly call any destructors needed to free up resources when the thread is terminated. If one thread cancels another thread, then it is not possible to automatically free up memory allocated by the canceled thread. The applications must consider this case carefully and free up memory to avoid memory leaks, perhaps by registering exception handlers. The kernel state of any semaphores held by a terminated thread is reset.

## Thread Cancellation Services

AIX provides the ability to cancel (that is, to terminate) threads while they are executing in a controlled fashion. For this purpose, AIX defines cancel points, typically provided when a function may block indefinitely. For instance, a `read()` system call may block forever; therefore, it is a cancel point.

**AIX allows resources to be released when a thread is terminated.**

AIX specifies a list of functions that are cancel points in its pthreads library and the standard C (thread-safe) library. A thread can set two states of cancel: cancel enable or cancel disable. A thread can also set two cancel types: deferred or asynchronous cancellation.

The cancel state can allow or disallow cancellation of a thread independent of its cancel type setting. If the cancel state is set to enable, the cancel type setting determines what happens when a cancel is posted on the thread. If the cancel state is enabled and cancel type is set to asynchronous, the thread is canceled immediately when a cancel is delivered to it. If the cancel state is enabled and cancel type is set to deferred, the thread is canceled only if it is waiting at a cancel point.

AIX also defines APIs to establish cancel cleanup handlers so that when a thread is canceled, it can clean up its state. Whenever a thread terminates, the registered cancel cleanup handlers are invoked.

OS/2 provides a simpler cancel interface in `DosKillThread()`, which terminates any thread other than the current thread. Any thread that is terminated can install an exception handler to capture the termination of the thread and perform any needed cleanup.

## Thread-Cleanup

AIX allows resources to be released when a thread is terminated. Thread-specific data can be cleaned up (released) by registering thread-specific data destructor routines when creating a thread-specific data key. Thread cancel cleanup handlers allow a thread to release mutexes and other resources held while executing when threads are canceled. AIX also allows a process to register exit time cleanup handlers.

In OS/2, a process can be terminated through a call to `DosKillProcess()`. In addition, a thread can terminate another thread in the process through `DosKillThread()`. A thread can terminate itself through a call to `DosExit()`. In all these cases, resources held by the application may need to be cleaned up. To allow this, OS/2 raises appropriate exceptions. To clean up semaphores in OS/2, each thread must have an exception handler to clean up during process termination (`XCPT_PROCESS_TERMINATE` or `XCPT_ASYNC_PROCESS_TERMINATE`). Alternately, `DosExitList()` can be used to add a cleanup handler at exit time for the process. If a process owing a mutex semaphore terminates without releasing a mutex, any other process that tries to request the mutex will get the error `ERROR_SEM_OWNER_DIED`.

Cleanup can be done by calling `DosQueryMutexSem()` on it to determine which process died, and then reinitializing appropriately. When a thread terminates itself through `DosExit()`, the `XCPT_PROCESS_TERMINATE` exception is generated. When a thread terminates another thread through `DosKillThread()`, the `XCPT_ASYNC_PROCESS_TERMINATE` exception is generated in the terminated thread. These exceptions provide opportunities for cleaning up resources held by the affected threads.

## Signal/Exceptions

AIX multithreading offers the `sigwait()` API that allows a dedicated thread to be set up to wait for asynchronous signals. This prevents undue interruption of application threads. A `sigwait` thread can receive the signal, then arrange for further synchronization with other threads through the mutex/condition variable APIs.

OS/2 does not have native support for `sigwait()`. Instead, it allows asynchronous signals (control C or control Break) to be

delivered to the main or initial thread (thread ID 1). Exception handlers can be set up in the main thread to act on these signal exceptions (signals converted to exceptions).

OS/2 supports the concept of deferring delivery asynchronous signals to the current thread through the concept of "must complete sections." An application can then protect itself from asynchronous exceptions during critical operations that need to mask asynchronous exceptions. Note, however, that synchronous exceptions are not deferred by this mechanism; the application should arrange to handle these synchronous exceptions.

## Timers

Both AIX and OS/2 support process-wide timers. When a thread sets up a timer, the expiration of the timer will be reported to the process as a whole, not to the individual thread that set up the timer.

However, AIX and OS/2 differ in the way they communicate the timer expiration. AIX sends a signal (`SIGALRM` or `SIGVIRT`) , whereas OS/2 posts an event semaphore associated with the timer.

## Appendix

Figure 2 shows the direct mapping of AIX and OS/2 thread APIs. Not all the AIX thread

```
int pthread_attr_setstacksize(pthread_attr_t *, size_t)
DosCreateThread()

int pthread_attr_getstacksize(const pthread_attr_t *, size_t *)
DosGetInfoBlock()

int pthread_attr_setschedpolicy(pthread_attr_t *, int)
DosSetPriority(ULONG scope, ULONG ulClass, LONG  delta, ULONG PorTid)

int pthread_attr_getschedpolicy(const pthread_attr_t *, int *)
DosGetInfoBlock()

int pthread_attr_setschedparam(pthread_attr_t *,const struct sched_param *)
DosSetPriority(ULONG scope, ULONG ulClass, LONG  delta, ULONG PorTid)

int pthread_attr_getschedparam( const pthread_attr_t *, struct sched_param *)
DosGetInfoBlock()

pthread_t pthread_self(void)
DosGetInfoBlocks()

int pthread_create(pthread_t *, const pthread_attr_t *, void *(*)(void *), void *)
DosCreateThread(PTID ptid, PFNTHREAD pfn, ULONG param, ULONG flag, ULONG cbStack)

int pthread_join(pthread_t, void **)
DosWaitThread(PTID ptid, ULONG option)

void pthread_exit(void *)
DosExit(EXIT_THREAD, ULONG result))

void pthread_yield(void)
DosSleep(ULONG msec)
```

*(continued on page 22)*

**Figure 2. AIX and OS/2 APIs that correspond to each other (OS/2 APIs are set in bold type).**

*(continued from page 21)*

```
int pthread_delay_np(struct timespec *)
DosSleep(ULONG msec)

int pthread_equal(pthread_t, pthread_t)
No Direct Support, but the thread ids can be compared since they are long unsigned
integers.

int pthread_getunique_np(pthread_t *, int *)
DosGetInfoBlocks() gives a structure that contains the thread id.

int pthread_setschedparam(DosSetPriority(ULONG scope, pthread_t,int,const struct
sched_param *)
DosSetPriority(ULONG scope, ULONG ulClass, LONG  delta, ULONG PorTid)

int pthread_getschedparam( pthread_t, int *, struct sched_param *)
DosGetInfoBlocks()  gives a structure that contains the priority class and level that
correspond with the scheduling class and priority respectively.

void pthread_cleanup_push( void (*)(void *), void *)
DosSetExceptionHandler( PEXCEPTIONREGISTRATIONRECORD pERegRec)

void pthread_cleanup_pop(int)
DosUnsetExceptionHandler( PEXCEPTIONREGISTRATIONRECORD pERegRec)

DosUnwindException( PEXCEPTIONREGISTRATIONRECORD phandler, PVOID pTargetIP,
PEXCEPTIONREPORTRECORD pERepRec)

int pthread_cancel(pthread_t)
DosKillThread(TID tid)

int pthread_mutexattr_setpshared( const pthread_mutexattr_t *, int)
(AIX4 does not support)
DosCreateMutexSem ( PSZ pszName, PHMTX phmtx, ULONG flAttr, BOOL32 fState)

int pthread_mutex_init( pthread_mutex_t *, pthread_mutexattr_t *)
DosCreateMutexSem ( PSZ pszName, PHMTX phmtx, ULONG flAttr, BOOL32 fState)

int pthread_mutex_destroy( pthread_mutex_t *)
DosCloseMutexSem (HMTX hmtx)

int pthread_mutex_lock(pthread_mutex_t *)
DosRequestMutexSem (HMTX hmtx, ULONG ulTimeout)

int pthread_mutex_trylock(pthread_mutex_t *)
DosRequestMutexSem (HMTX hmtx, 0)

int pthread_mutex_unlock(pthread_mutex_t *)
DosReleaseMutexSem (HMTX hmtx)
```

*(continued on page 23)*

**Figure 2 (continued). AIX and OS/2 APIs that correspond to each other (OS/2 APIs are set in bold type)**

*(continued from page 22)*

```
int pthread_mutex_setprioceiling(pthread_mutex_t *, int, int *)
(AIX4 does not support)
None

int pthread_mutex_getprioceiling(pthread_mutex_t *, int *)
(AIX4 does not support)
None

pthread_mutex_getowner_np(pthread_mutex_t *mutex)
DosQueryMutexSem (HMTX hmtx, PID *ppid, TID *ptid, PULONG pulCount)

int pthread_key_create( pthread_key_t *, void (*)(void *))
DosAllocThreadLocalMemory( ULONG cb, PULONG *p)

int pthread_once( pthread_once_t *, void (*)(void))
DosExitCritSec(VOID), DosEnterCritSec(VOID)

int sigthreadmask(int how, const sigset_t *set, sigset_t *oset)
DosEnterMustComplete( PULONG pulNesting)
      DosExitMustComplete( PULONG pulNesting)

int pthread_kill(pthread_t, int)
DosRaiseException( PEXCEPTIONREPORTRECORD pexcept)
DosSendSignalException(PID pid, ULONG exception)
```

**Figure 2 (continued). AIX and OS/2 APIs that correspond to each other (OS/2 APIs are set in bold type)**

interfaces have corresponding OS/2 APIs, and not all the OS/2 APIs have matching AIX interfaces. Figure 2 shows APIs that correspond to each other. Those APIs not listed here do not have direct mapping. Since AIX 4 does not support interprocess shared mutexes and condition variables, porting applications in OS/2 that depend on these facilities is a challenge. It is possible to use other interprocess communications facilities in AIX 4 to obtain an equivalent functionality.

**Chary G. Tamirisa**, IBM Corporation, 11400 Burnet Road, Austin, TX 78758. Internet: chary@austin.ibm.com. Mr. Tamirisa was the team leader for the threads package on AIX and OS/2 DCE. He has also worked in the fields of communication protocols, system software, and National Language Support. Mr. Tamirisa has an MS in Computer Science from McGill University and a BTech in Electrical Engineering from the Indian Institute of Technology in Madras, India.

# RS/6000 Systems Support Lotus Notes Release 4

Lotus Notes® Release 4 is supported on the RISC System/6000 (RS/6000) family of workstations and servers running the advanced AIX operating system.

Lotus Notes is already the world's leading messaging and groupware product. The increased capacity for users provided by Release 4 will require larger Notes servers. Release 4 will also have the ability to run more critical business applications on Notes than ever before. The IBM RS/6000 provides the blend of scalability, manageability, and availability necessary for such an enterprise solution.

The RS/6000 family ranges from workstations to high-performance servers supporting a comprehensive array of system and network management software to ease the complexity and cost of administering enterprise server environments. In addition, the RS/6000 can be configured to provide the high availability required for mission-critical Notes applications.

The following are ways in which the RS/6000 can be used to provide a true enterprise environment for Lotus Notes Release 4 users:

## Scalability

- A wide choice in capacity of RS/6000 systems allows users to match system price/performance to their Notes needs.
- High-end uniprocessor, symmetric multiprocessor, and Scalable POWERparallel™ (SP) systems models provide room for future growth of Notes systems.
- As users grow their systems, they retain their Notes software investments, since the same AIX operating system runs across the entire RS/6000 line.

## Manageability

- The processing power of the RS/6000 can be used to consolidate—on a single RS/6000 server—a Notes workload that was previously distributed across multiple servers, making it easier and more cost-effective to manage.
- If multiple servers are desired to segregate Notes workload, the systems and network management software of the IBM SystemView® family helps Notes users to more easily administer this network of systems.
- Users can leverage both scenarios by combining multiple individual Notes servers within an IBM RS/6000 SP complex, providing a single-system image for systems management while retaining the availability and workload balancing benefits of individual servers.

## Availability

- RS/6000 Notes servers can be configured in clusters using IBM High Availability Cluster Multiprocessor (HACMP) software to ensure that if one system fails, its work is taken over by another, without incurring the added expense of fully redundant hardware.
- HACMP clusters can even be used within an SP complex to provide a high-availability environment for one of the SP servers.
- Availability also means providing connectivity to Notes clients existing on PCs in various LAN environments.
- The RS/6000 supports a wide range of PC/LAN hardware and software connectivity options to tie an RS/6000 Notes Release 4 server in with existing clients.
- To make Notes applications available to an even wider audience, the Lotus InterNotes™ Web Publisher can be used in conjunction with World Wide Web (WWW) server products, such as the IBM Internet Connection Server for AIX, to allow Internet access to Notes and Notes databases.

### For More Information

See the following Internet home pages:

| | |
|---|---|
| IBM Home Page | http://www.ibm.com. |
| IBM RS/6000 Home Page | http://www.rs6000.ibm.com. |
| IBM Lotus Home Page | http://www.lotus.com. |

Call the IBM Fax Information Service to receive facsimiles of IBM product press releases. Simply dial 1-800-IBM-4FAX and

- The RS/6000 can also provide links between Notes applications and existing applications, such as those based on DB2® and CICS™ for example, using optional software products.

# TPC-D—For Comparing Performance of Decision-Support Systems

IBM has published the first audited results of the new TPC-D industry-standard benchmark. This benchmark is a suite of real world, business-oriented database queries and updates that are designed to simulate decision-support operations on a computer system.

Benchmarks are tests used to gauge the performance of a computer system and enable comparisons between different systems. Benchmarks can be created to evaluate how well a computer system will perform specific tasks or functions. The TPC-D benchmark was established in April of 1995 by the Transaction Processing Performance Council, an independent benchmark standards body. It focuses on how systems examine large volumes of data and perform complex analyses to provide answers to critical business questions. This technique is commonly referred to as decision support, since it is useful in guiding business decisions to gain a competitive advantage.

These initial TPC-D benchmarks were run on an IBM RISC System/6000 (RS/6000) Scalable POWERparallel (SP) System using the IBM DB2 Parallel Edition database and IBM 7133 Serial Storage Architecture (SSA) disk storage devices. This combination provides a high-performance, scalable platform for decision-support applications.

The extensive processing power made available by the RS/6000 SP today supports advanced analysis techniques such as data mining, in addition to classic decision support, to gain maximum value from enterprise data. The RS/6000 SP system is a member of the RS/6000 family of RISC-based workstations and servers running AIX.

DB2 Parallel Edition for AIX is designed to run complex queries on very large databases, enabling decision-support and data mining applications. DB2 Parallel Edition takes full advantage of the RS/6000 SP's architecture where each processor has its own operating system, internal memory, and disk. DB2 Parallel Edition performance increases in a near linear fashion as data is distributed incrementally across the RS/6000 SP's multiple processors. As a result, as more data is acquired, query response times can be maintained by adding processor nodes to the RS/6000 SP.

Decision-support systems typically involve large quantities of data. The IBM 7133 SSA disk subsystem improves throughput and availability of database applications involving large numbers of queries against massive data stores. Performance features include dual-path connectivity running 80 MB/sec per loop, four times faster than SCSI-2-based storage. AIX mirroring techniques combine with the SSA design to ensure high availability. SSA keeps critical data flowing to customers and users with redundant data paths to preclude any single point of failure.

These benchmark results, based on the initial threshold of 100 GB of data, were reported for the TPC-D's four requisite components:

- QppD@100 GB = 207.01 (TPC-D Power metric)
- QthD@100 GB = 84.58 (TPC-D Throughput metric)
- Price/Performance = $33,640.49 per QphD@100 GB (TPC-D price/performance metric)
- Availability Date = present

Additional TPC-D benchmark tests with significantly more data are planned. The TPC-D results, along with other important characteristics such as reliability and availability, will provide data to compare various decision-support systems and their capabilities.

# IBM Solution Developer Program

The IBM Solution Developer Program is a worldwide IBM initiative to help commercial solution developers grow their business on IBM software and hardware platforms and technologies. It is designed to simplify access to a wide range of IBM development support programs by providing a single coordinating point for developers with IBM. The IBM Solution Developer Program combines and improves existing development offerings from the various IBM product divisions into one worldwide program.

The IBM Solution Developer Program provides solution developers with a consistent and reliable set of offerings and information on technical, business, and marketing support services. It capitalizes on the focus of today's specialty offerings which are called Partners in Development such as the OS/2 Partners in Development, AS/400® Partners in Development, and Object Connection Partners in Development.

The IBM Solution Developer Program covers all IBM operating systems (OS/2, AIX, AS/400, and MVS®), software products such as DB2, CICS, NetView®, and specific technology areas such as object technology and systems management. The program will be especially useful to commercial software developers who want to build mission-critical, multiplatform, client/server applications.

There are two types of membership:

- Commercial Developer Member for companies who develop software for the commercial market
- Individual Member for people registering as individuals

Some of the program content is clearly designed to help companies expand their market opportunity and will only be available to Commercial Developer Members.

As a member, you will find IBM strategy documents, technical insights, and other information


**Visit the Solution Developer World Wide Web page for access to information and developer services**

that is critical to you and your business if you are to expand into existing and emerging markets.

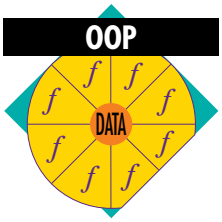
The IBM Solution Developer Program can provide the services you have requested:

- Easier contact and consistent ways to do business with IBM
- A complimentary information tier with access to specialty offerings designed to help you grow your business through successful development on IBM technologies
- A worldwide and standardized set of deliverables
- A single, universal membership number and password that identifies you as a member across all IBM development platforms
- Easy access to information and Solution Development Program Services through the World Wide Web

## For More Information

| | |
|---|---|
| E-mail | ibmsdp@vnet.ibm.com |
| Solution Developer Program (fax) | 1-770-835-9444 |
| Solution Developer Hotline (worldwide) | 1-770-835-9902 |
| Solution Developer Hotline (U.S. and Canada) | 1-800-627-8363 |

# Choosing an Object-Oriented Analysis and Design Tool




By Michael J. Branson and Eric Herness

Successfully deploying object-oriented analysis and design tools requires a full understanding of the requirements and capabilities of the tools. This article provides a comprehensive list of these requirements. Scaling up to team environments is critical to successful deployment of this technology in large organizations. A rich and robust object-oriented analysis and design tool that works well in a team environment will help to ensure a consistent application of the object-oriented development process across the entire software development team.

Selecting the right tools for software development projects can be difficult. Every project has unique needs, and there are many different tools from which to choose. One category of tool selection is particularly challenging—choosing an object-oriented analysis and design tool—because there are so many issues to consider, such as methodology, scalability, and integration. This article details a set of requirements for object-oriented analysis and design tools and classifies those requirements into areas important in your software development projects.

Although each project is unique, all object-oriented development projects share some common set of needs. Figure 1 illustrates the different types of requirements that projects can have on object-oriented analysis and design tools. It also shows the relationship between the different types of tool requirements and certain project attributes.

The 'x' axis in Figure 1 depicts the degree to which projects require the analysis and design tool to pervade the object-oriented life cycle. Is the tool limited to covering iteration through analysis and design activities, or does it also cover creating and maintaining code? The 'y' axis represents project size since the importance of various analysis and design tool requirements depends upon the number of developers involved in the project.

The remainder of this article describes various requirements in each category, including detailed explanations for each requirement.

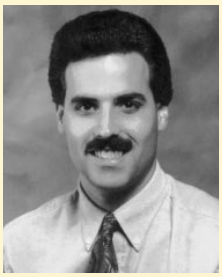
## Methodology Requirements

Methodology requirements deal with support provided in the tool for the object-oriented paradigm. Developers use the tool to explore the software domain of the project and design a suitable implementation. An object-oriented analysis and design tool must allow developers to practice a software development method that leverages the features of object-oriented programming, such as abstraction, encapsulation, inheritance, and polymorphism.

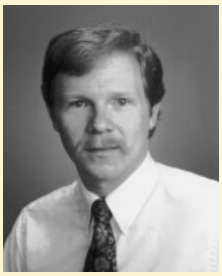
All projects, regardless of size, require a methodology. Our experience shows that the exact methodology chosen is not nearly as important as having the appropriate support for the chosen method. Tools, education, and processes must work together with the chosen method and be adapted to meet the needs of software developers in the most expedient way.[1] Therefore, methodology requirements are an issue for all projects when choosing an object-oriented analysis and design tool. The key methodology requirements are described in the following sections.

### Object-oriented Method and Notation

An *object-oriented method* is a disciplined process for generating a set of models describing various



*Michael J. Branson*



*Eric Herness*

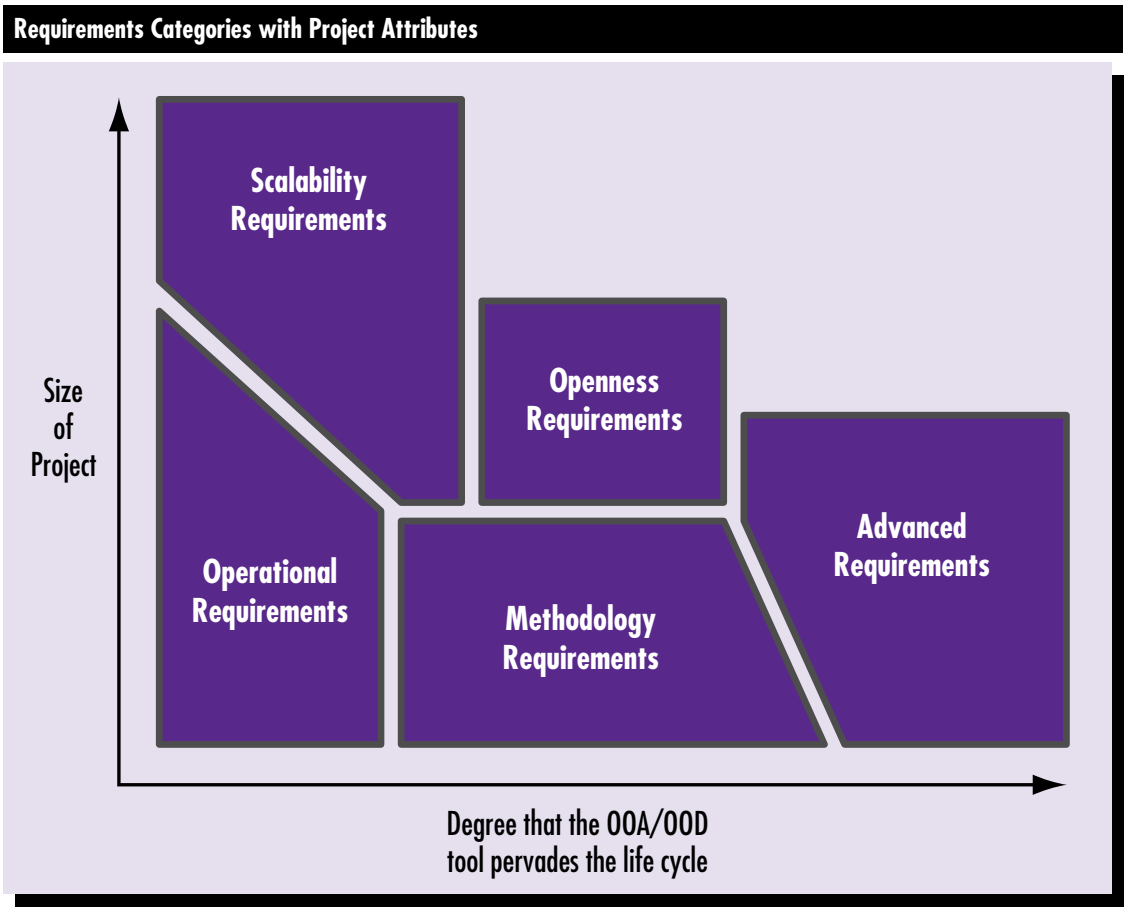



**Figure 1. Positioning of requirements categories with project attributes**

aspects of the software system under development, using some well-defined notation.[2]

Selecting the method to use for your project is an important decision. The tool used for analysis and design should support the chosen method. It is not effective to have developers thinking within the context of a particular object-oriented method, then translating those thoughts into models that are not part of that method and notation. The tool must map to the intellectual concepts with which developers have chosen to work.

Many object-oriented methods and notations exist.[2, 3, 4, 5] Choosing a well-known method and notation allows software developers and project managers to seek support services including literature, education, and consultation from many sources. Those practicing software development using a popular method can leverage a larger marketplace of ideas that exists in the method's user community.

Examples of AIX-based tools that support a particular method include Rational Rose® from Rational® Software Corporation that supports the Booch method, and Objectory® SE from Objectory Corporation that supports the Jacobson use case-driven approach to object-oriented software development. Also, a class of meta-tools that are not tied to a particular method allow you to define the notation that you want supported. Although these tools provide greater flexibility, they require customization to meet all the requirements met by method-specific products.

Frequently, tools only support part of the chosen method. Be sure that when a tool is advertised to support a particular notation, it supports the notation fully—or at least to the extent that your project needs to leverage it.

### Legal Uses of the Notation

Legal uses of the notation must be enforced. If an analysis and design tool allows misuse of a notation, then it is simply a drawing tool. Using the tool, you should be able to construct models that are consistent with the notation and prohibit illegal use of this notation. The tool should

enforce the syntactic correctness of models produced. Proper use of the notation allows projects to experience the benefits offered by the method's rigor.

### Static Model of Classes

The methodology should support a static model of classes and their relationships. All object-oriented methods have some notion of a static model of the software system. Booch, for example, uses class diagrams.[2] The exact content of this static model varies from method to method.

The static model depicts entities (usually classes of objects) and relationships between them. Relationships minimally include association, aggregation, and inheritance. Whatever the notation and method, the static model is essential to the object-oriented analysis and design methodology.

### Class Attributes and Behavior

Static entities require more specification than merely naming. The key attributes and behavior of the entity should be documented as part of analysis; implementation language-specific characteristics of a class should be documented during the design activity. Some tools may not allow for such specifications; others may include specifications, but may be limited to only language-independent characteristics. The detail allowed by the specifications determines how long the developers can use the tool before they must move to other forms of specifications, such as the implementation language itself. This is often the gating factor that determines whether the tool can generate reasonable code.

### Dynamic Object Model

A dynamic model of objects and their interactions with one another is often necessary to accomplish a particular scenario. Most methods support one or more types of dynamic models. For example, Booch[2] supports object diagrams that show objects and their relationships in the logical design of a system. Booch[2], Rumbaugh[3], and Jacobson[4] support interaction diagrams that provide specific support for tracing the execution of a scenario in the system being modeled. Dynamic models are constructed to validate the static model of the system. Each scenario tests whether the evolving static model is complete and optimal for that portion of the system's dynamic behavior. Support for dynamic models is another essential element of an object-oriented analysis and design tool.

### User-Oriented Scenario

End-user scenarios are handled differently in some methods. If your method relies upon user-oriented scenarios such as use cases[4], the analysis and design tool should support them.

### Dynamic State of Object

The state information of some objects is so complex that it is often essential to use a tool to model the states and state transitions of the object. This is especially evident in real-time software objects and some user interface objects. An object-oriented analysis and design tool should provide the capability for state models. If your project will not include objects with complex state transitions, you may be able to use a tool that does not provide support for a state model.

When leveraging state models, look for a tool that supports nested states based upon the work of Harel[7]. Modeling with simple states often leads to very complex diagrams. Nested states make it easier to model complex state transitions.

**Any analysis and design tool should contain certain operational requirements that make the tool practical or easy to use.**

### Architectural System Model

System architecture is important in any software undertaking. The system must be designed at the subsystem level as well as the object and class level. Although architectural diagrams will vary from method to method, support for them is very important. Look for a design tool that will enforce the architectural decisions set forth in much the same way that the notation might be enforced. The tool should ensure that the architected and exported interfaces to a subsystem are known. It should also ensure that classes internal to a subsystem are not used by objects outside that subsystem.

### Physical Model

The tool should support a physical model (module view) of the software, the relationships between modules, and the allocation of modules to processors. So far, all the requirements have focused upon constructing a logical model of the software system. One step in every object-oriented development method is making a physical model from a logical model. This involves decisions about object and class distribution across modules, shared libraries, tasks, and threads. In simple situations such as a single-threaded software system that runs completely on a single machine, this mapping may be trivial and may not require modeling. But in distributed systems and multiprocessing

environments, good support for capturing the physical model is essential.

## Operational Requirements

Besides supporting the development method, object-oriented analysis and design tools must provide a base set of functionality, not related to methodology. Any analysis and design tool should contain certain operational requirements that make the tool practical or easy to use. Software developers rely on these operational requirements to accomplish activities that are not necessarily defined by their object-oriented development method. Operational requirements are described below.

### Item Dictionary

The tool should support an underlying dictionary of the items used in models. The dictionary, or repository, that holds the definitions of the items in the diagrams should be available for reference or usage on similar items in other diagrams. A reliable test of dictionary support is whether you ever need to type the same information twice. A dictionary also enables multiple views of the same underlying data. Multiple views are important when presenting material about designs to various participants in the development process.

**The primary purpose of object-oriented analysis is to explore the software system domain that you are building.**

### Semantic Checking for Inconsistencies

The requirement for semantic checking between models or diagrams for inconsistencies relies upon the dictionary mentioned above. The tool should alert you to contradictions in a model. For example, if a class has been characterized as an abstract class in one diagram or specification, while in another diagram an object is being instantiated from it, you should be aware of this conflict.

### Printing in Standard Format

Printing is an essential requirement. PostScript® printing support generally makes the tool usable in many environments. The ability to export data from the tool in a variety of formats is also desirable, with specifics being dependent upon the underlying operating system.

### Generating Design Documents

The tool should allow you to generate design documents from collections of models and other design information. The primary purpose of object-oriented analysis is to explore the software system domain that you are building. The next step may be communicating what you have learned to others in the project; therefore, the tool's ability to communicate results of an analysis is an important feature. It is not enough to simply generate a good analysis; we need to share it with other people, and potentially with other tools as well. The results of design activity have the same requirements.

One inherent attribute of medium-to-large software development projects is the large amount of data to assimilate. Choosing a tool that packages and organizes the models created in analysis and design into a coherent document or set of documents is important. Without this support, these documents may need to be hand-crafted. They may be combined in a way that does not communicate well. It is imperative that an analysis and design tool make generating documents a push-button activity, which encourages more exploration and less document preparation.

### Platform Interoperability

In addition to generating a flexible set of documents, the analysis and design tool should integrate well with the underlying operating system platform. It should be easy to copy and paste portions of diagrams to other tools that operate on the platform. This enables you to quickly create ad hoc design packages and presentations. For example, if the word processor supports graphics image manipulation, then the tool should enable figures to be exported in the appropriate format. Hyperlinks and linking are also desirable characteristics because copying brings the risk of the material becoming outdated.

If there are tool integration facilities on your development platform, the analysis and design tool should integrate into the tools workbench. For example, in AIX Version 4, the capability to integrate the tool with desktops such as IBM's AIXwindows, which supports the Common Open Software Environment (COSE) consortium's Common Desktop Environment (CDE) specification, would provide interoperability at the user interface level. Desktop integration alone is not enough. The capability to integrate via underlying messaging mechanisms like ToolTalk® is necessary to gain true interoperability between the analysis and design tool and other tools in the environment.

The tool should also interoperate among platforms. If the tools run on various operating systems (AIX, OS/2, Windows™), design information should be easily exchanged among platforms. This is the first step toward supporting multiple and mobile users. This topic will be

explored in detail in the section entitled "Scalability Requirements."

### Class Interfaces and Source Code

The tool should allow you to view and edit class interfaces and source code from models. In an iterative development activity, the ability to move quickly between the artifacts of analysis, design, implementation, and testing is important. When working within an analysis and design tool, the ability to select a class and see any source code that exists to implement the class is helpful.

Another related key feature includes the ability to view the source code using the developer's chosen editor. This implies a level of integration with other tools. Finding the source code from the design tool must be done correctly. It must use the same information to find source code that build or make tools would use to find source code during a compile.

### Filters for Multiple Views

The tool should have the ability to apply filters to a particular model that allows multiple views (for example, inheritance only) of the same model. Another operational characteristic deals with the presentation of the information contained in the dictionary. First, the ability to get an inheritance-only or an aggregation-only view is important. Sometimes the density of a diagram is very high. While this is appealing to experienced designers, other consumers of design information might like a gentler introduction to system analysis and design.

Another filter example is one that deals with public versus private. Diagrams may contain various relationships and entities that are private. It is sometimes desirable to take a public-only view.

### Model Management

A final operational characteristic is *model management*, the ability to evolve a system without requiring hours of tools housekeeping. For example, the movement of a class from one subsystem to another must be accomplished with ease. If there are objects instantiated from this class that is shown on many other diagrams, one simple command should move it from one system to another and update all other references to this class.

The same must happen at the subsystem level. Analysis and design often start with a few rudimentary subsystems, then proceed with capturing many classes and objects. Refinement of the architecture may require additional subsystems as well as modifications to the nesting of subsystems—a natural occurrence in most large projects. This type of iteration and evolution must be easily accomplished in an analysis and design tool.

## Scalability Requirements

In order to support medium to large projects, tools must scale both vertically and horizontally. Vertical tools should be usable by all levels of the organization. Abstract views should be available for technical management and detailed views available for developers. Horizontal scaling of the tool refers to the size of the project and the number of people who will be using it.

### Scalable to Large Projects

Tools must be scalable to support large projects. Each organization must determine the size of projects to be supported by a particular analysis and design tool. A good measure for this is the total number of classes expected in the systems to be constructed. It is important that the tool selected has been previously used in other projects of comparable scale.

Often tools differ in their definition of a project. Some tools are constructed with a project defined as work to be done by a single developer. Tools that isolate one developer's analysis and design work from work by other developers of the software system have very limited use. Analyzing and designing a software system requires a tool whose notion of project equates with the notion of an entire software system.

**Analyzing and designing a software system requires a tool whose notion of project equates with the notion of an entire software system.**

### Multiple Users on Same Project

The tool should allow multiple users to work simultaneously on the same project, or allow for integration with the configuration management system in the development environment. Once again, the correct interpretation of the term "project" is important. If a tool scales well enough to allow the entire analysis and design of a software system to be done in one project, this presents the problem of how multiple developers work on the same project at the same time. Many analysis and design tools are single user. With a sufficient degree of openness, a single-user tool integrated with a configuration management system can behave like a multi-user tool.[6]

We believe that the *open* option is the right one. Analysis and design tool vendors are not generally focused on providing extensible and scalable configuration management support. A

better option for them is to provide the 'hooks' that allow configuration management tools to be the repository for design information. Multiple users can then check in and check out various design units. We have successfully used the CMVC/6000 product on AIX to support multiple users of a single-user analysis and design tool.

The next issue is the number of units of granularity supported in the tool, which will be a key determinant for configuration management integration. Our experience indicates that the subsystem is a reasonable and sufficient level of granularity for most projects. This is based on using C++ and having subsystems that contain 5 to 50 classes. Your results may vary. Although the configuration management overhead associated with having class-level granularity is substantial, this is a reasonable requirement for domains where strict ownership and control are needed.

**Since every software project is somewhat different, minor modifications to the analysis and design tool can be critical to ensuring reasonable adoption of the tool.**

### Maintain Different Versions of Model

A final scalabilty requirement deals with model versioning. The tool must be able to maintain different versions of a model or allow integration with the CM system to meet this need. Large projects will build systems in several increments.[8] Each increment will have an analysis and design model associated with it in addition to the code that is built. The iterative and incremental nature of the object-oriented development process dictates that multiple versions of the system will be analyzed and designed concurrently. One team may be designing increment *n*, while another is coding and testing increment *n-1*. Both require access to a 'version' of the analysis and design model that matches the code associated with the increment being worked. This requirement can be met by having a checkpointing or 'model freezing' facility in the tool or enabled by integrating the tool with the CM system used for the code.

## Openness and Extensibility

In general, selected analysis and design tools should be extensible. Since every software project is somewhat different, minor modifications to the analysis and design tool can be critical to ensuring reasonable adoption of the tool. This section highlights two important 'openness and extensibility' items. It does not detail basic items such as add-on menus, callable Application Programming Interfaces (APIs) to the data dictionary, export formats, and user-interface preferences or resources.

### Linking

Analysis and design models can be linked with other tools. Traceability is often required in object-oriented software development. In an object-oriented project, requirements can be linked to scenarios that can be linked to the dynamic model that illustrates them and the test cases that validate them. Items on models can also be linked to the source code that implements them. Many more possibilities exist for linking to support traceability.

Linking items in an analysis and design tool with items such as requirements, use cases, or code outside the tool requires openness. The tool should allow dictionary entries for an item to be extended. It should also allow the addition of functionality to the tool that uses this information to link to items from outside the tool or other tools from within the analysis and design tool.

### Add-On Function Support

There may be more to be accomplished in a project's development process than what your chosen object-oriented development method identifies. A tool that supports the method may also need to support additional activities specific to your project's development process. To provide this support, the tool must offer features such as user-defined menus, and open its dictionary to the data used by these add-on functions.

It is not efficient to add menus to a tool if the data associated with these add-ons must be managed in a separate environment. For example, if a reuse ID was associated with some classes produced by your development organization, the ability to store that information with the rest of the data for that class would be helpful.

## Advanced Requirements

Advanced requirements contains functions not found in most of today's analysis and design tools, or functions that are present in some, but still need revision. Although the lack of availability of some of these attributes should not cause you to reject a candidate tool, the presence of some of these capabilities might influence your decision-making. If the analysis and design tool supports these requirements, you can exploit the object-oriented paradigm more fully and use the tool later in the object-oriented life cycle.

### Model Metrics

The design tool should be able to output certain metrics (such as number of classes, number of

changed classes, number of methods) about the models in a project. It might output these metrics or allow this type of information to be extracted from its dictionary. Metrics are an important project management tool. Today, some organizations pull object-oriented metrics from the source code. However, it is just as valid to measure these values in analysis and design models to make mid-project adjustments to the way the object model is being done.

### Design Integration

The design tool should be able to distinguish between the design commitment or integration to assist metrics in distinguishing "real" design from early illustrations that have been put aside. The capability to mark or select classes that are system components should be part of checkpointing a design. Some tools will support identifying 'obsolete units' while others might have a reporting facility that will match classes defined on a static model against usages on dynamic models. Proper integration to the configuration management system can also help facilitate a clean 'build' of the design. The ability to populate a model from existing code and compare it to an existing model might also support finding the real 'design.'

### Class Interfaces from Specifications

If the tool allows class information to be specified at a sufficient level of detail, it should be able to generate class interfaces (code and documentation) from specifications. The tool does not magically add any information to the interface when going through the generation step; however, it does generate code that reflects the specifications consistently. All class interfaces produced through generation may look and follow the same generation standard.

### Code Generation Parameters

If a tool generates code only in one fixed format, its generation feature is probably not worth using. If the tool requires hand-crafting the code after it is generated, there is little value in the code generation support.

The tool should allow you to customize code generation parameters. This will enable you to do simple tasks such as controlling the names of the files produced by the generator, controlling where documentation is placed in the interface, or controlling advanced attributes that are tied to the version of your implementation language and your project's chosen programming style. It is also important to ensure that the code generation support works with the compiler being used for the project.

### Static Diagrams from Source Code

A valuable attribute of an analysis and design tool is the ability to generate diagrams statically from source code; that is, reverse engineer static diagrams from source code. This can help you understand a software system without a well-documented design. Another use may be to understand what was actually implemented in a project.

This capability can also produce models of reusable class libraries or frameworks. These models can then be brought into the design tool and used in projects to tie the analysis and design of the software system to used class libraries and frameworks.

### Comparison of Generated and Manually Created Diagrams

Generating diagrams creates more data, which must be easily compared to the analysis and design model. The ability to compare the differences in a generated diagram and a manually created one needs additional support in most tools today. It is not sufficient to 'eyeball' a design diagram in one window and a generated diagram in another.

**A valuable attribute of an analysis and design tool is the ability to generate diagrams statically from source code.**

### State Model

Another requirement is the ability to execute the state model. If a tool has allowed you to specify the complex state transitions of an object, it may also allow simulation of the state model. This capability may require that you specify the state information for all objects in the system, which may only be practical for real-time or other state-intensive system development.

### Implementation Code from State Model

You should be able to generate implementation code from the state model. Relating to the previous requirement, implementation code generation can also be supported where detailed state information has been gathered. Unlike the execution or simulation requirement, code can be generated only for those objects that have state diagrams.

## Conclusion

Many requirements exist for object-oriented analysis and design tools using the definitions of those tools prevalent today. As we move into the future, various tools will be integrated and distinctions between the types of tools will blur. As code generation technology emerges and object-oriented frameworks and domain class libraries become more pervasive, the tasks that developers expect analysis and design tools to accomplish will broaden and evolve. More time will be spent exploring (looking for potentially reusable units) and interrogating (testing a reusable component to evaluate its ability to work in a particular situation) to ensure a consistent application of the object-oriented development process across the entire software development team.

## References


1. Branson, Michael and Herness, Eric. "The Object-Oriented Development Process." *Object Magazine* (November-December 1993): 66-70.
2. Booch, Grady. *Object-Oriented Analysis and Design with Applications*. Redwood City, California: Benjamin/Cummings Publishing Company, Inc., 1994.
3. Rumbaugh, James; Blaha, Michael; Premerlani, William; Eddy, Frederick; and Lorensen, William. *Object-Oriented Modeling and Design*. Englewood Cliffs, New Jersey: Prentice-Hall, 1991.
4. Jacobson, Ivar; Christerson, Magnus; Jonsson, Patrik; Overgaard, Gunnar. *Object-Oriented Software Engineering—A Use Case-Driven Approach*. Wokingham, England: Addison-Wesley Publishing Company, 1992.
5. Wirfs-Brock, Rebecca; Wilkerson, Brian; Wiener, Lauren. *Design Object-Oriented Software*. Englewood Cliffs, New Jersey: Prentice Hall, 1990.
6. Branson, Michael and Herness, Eric. "Building an OO Development Environment on AIX." *AIXpert* (February 1995): 35-44.
7. Harel, D. Statecharts. "A Visual Formalism for Complex Systems." *Science of Computer Programming*, Volume 8, 1987.
8. Branson, Michael and Herness, Eric. "Structure and Management of Object-Oriented Projects." *Object Magazine* (May 1994): 33-38.



**Michael J. Branson**, IBM Corporation, AS/400 Division, 3605 Highway 52 North, Rochester, MN 55901. Internet: mjb@chvmw3.vnet.ibm.com. Mr. Branson is an advisory programmer, tools strategist, and object-oriented consultant to internal projects in Rochester. He has a BS in Computer Science from Purdue University.

**Eric Herness**, IBM Corporation, Software Solutions Division, 3605 Highway 52 North, Rochester, MN 55901. Internet: herness@vnet.ibm.com. Mr. Herness is a senior programmer working in object-oriented systems development. He has a BS in Business Administration from the University of Wisconsin at Eau Claire and an MBA from the Carlson School of Management at the University of Minnesota.

# IBM AIX Support Line Basics


**By Georgia A. Gibson**

This article provides information about IBM AIX service offerings available online or by phone or fax. It also gives a thumbnail sketch of other IBM Support Family services.

When a company signs up for AIX Support Line, IBM sends documentation about how to use the service, the hours that support is available, and other necessary information. However, as people move to new jobs, the Support Line documentation often becomes lost, and the new staff member does not have the most current information. Then when a problem arises, it is unclear about what to do or expect from IBM.

AIX Support Line receives many calls each day. Some calls are from those who are new to Support Line; others have just recently become the authorized contact for Support Line. Still others have problems and are unfamiliar with Support Line.

This article provides information about IBM AIX service offerings available online or by phone or fax. It also gives a thumbnail sketch of other IBM Support Family services.

The AIX service offerings discussed in this article are available in the United States. If you are outside the U.S., refer to your local IBM Availability Services representative for your country-specific contact. If you have access to the Internet, you can obtain specific information that pertains to your country by doing the following:

- With your favorite browser, type www.ibm.com
- Select What We Offer
- Select Services
- Select Availability Services
- Select your country from Choose a Country
- Read the details

## Telephone Resources for Information

Several resources are available to obtain information about AIX and the RISC System/6000 (RS/6000). IBM provides several toll-free 800 numbers for you to learn about service offerings, obtain service, or order product information via fax.

### AIX Support Line Contract

AIX Support Line is a fee service. With your customer number and a brief description of your problem ready when you call, the response coordinator can quickly connect you to a technical specialist who can help. Use the phone numbers in Figure 1 for more information about the AIX Support Family of services.

### Other U.S. Services

To learn about more services in the U.S. (listed in section below), call AIX Availability Services at 1-800-IBM-4YOU. You can also call the AIX Support Family Project Office at 1-800-CALL-AIX (option 8), or send E-mail to callaix@vnet.ibm.com.

Georgia A. Gibson

| Phone Number | When to Call |
|---|---|
| 1-800-CALL-AIX (1-800-225-5249) (U.S. only) | Questions about AIX on the RISC System/6000 |
| 1-800-237-5511 (U.S. only) | Questions about other IBM platforms |
| 1-817-491-0053 | AIX Support Family information if you are not in the U. S. |
| 1-800-438-2468 | Order supplies related to your RISC System/6000 |
| 1-800-879-2755 | Order additional publications for your RISC System/6000 or AIX |

**Figure 1. AIX Support Family access numbers**

### IBM Fax Information Service

The IBM Fax Information Service is available 24 hours per day, seven days a week. You can call at any time of the day or night and request information to be sent to the fax machine of your choice. First time users should be sure to request the new user instructions by responding to the instructions on the phone.

The IBM Fax Information Service number is 1-800-IBM-4FAX (1-800-426-4329). If you know the specific document number, you can request that document by phone; for example, request document #1228 for FixDist or document #2441 for PitStop. Subscription and speed-dial services are also available via this number.

Other useful fax numbers include the following:

| | |
|---|---|
| 1-800-522-3422 | RISC System/6000 supplies |
| 1-800-879-2755 | RISC System/6000 or AIX publications |
| 1-415-855-4329 | To call the fax from outside the U.S. using a fax machine phone |

## Hardcopy Information

The AIX Support Family Welcome Package notebook is sent to you automatically when you sign up for AIX Support Line. It contains information about a variety of AIX Support Family Services available for purchase. If this book is not available at your location, call 1-800-CALL-AIX to request this package. It provides specific access and usage information about the following fee-based services:

**AIX Support Line Standard:** Service that provides telephone access to technical specialists who can help with general installation, usage, and code-related questions about RISC System/6000 hardware, AIX software, and Scalable POWERparallel™ systems

**AIX Support Line Premium:** An extended package of AIX Support Family services with discounts for other services and education courses

**AIX Associate:** Service that provides a remote support analyst who will act as your advocate on software installation, usage and defect issues, and hardware service

**AIX Technical Library:** Comprehensive AIX library of service and support information on CD-ROM; uses the user-friendly InfoExplorer hypertext search and retrieval facility

**AIX Alert:** Service that automatically notifies you by fax or Internet E-mail of code-related fixes that may help prevent problems

**AIX Consult Line:** Access to AIX consultants via scheduled conference calls for help in resolving questions that are beyond the scope of AIX Support Line, such as performance analysis, capacity planning, and high availability

**AIX/6000 Performance Management:** Service that assists you in understanding and managing system resource utilization through ongoing analysis of key performance indicators of your RISC System/6000

**AIX System Backup and Recovery/6000 (Sysback):** Sysback enables you to install, back up, list, verify, and restore various types of data on an RISC System/6000 to both local and remote devices

**RISC System/6000 Recovery Express:** Disaster recovery service that provides a backup system to you within 24 hours of a disaster

**IBM House Call:** On-site assistance with tasks such as software installation, problem assistance, and Program Temporary Fix (PTF) updates

The Welcome Packet also contains an IBM Support Family brochure and the latest issue of */AIXtra*, IBM's magazine for AIX professionals.

## Service Tools

Two service tools are important for providing information or code to keep your system current.

### AIX/PitStop

AIX/PitStop is a prototype AIX desktop service and support facility that all AIX customers receive (You do not have to subscribe to the AIX Support Line). AIX/PitStop was designed to make existing tools and information easy to access using one common user interface.

AIX/PitStop can help you with the following functions:

- Accessing information with or without the Internet
- Reading all of your README files by logically grouping them together
- Subscribing to the latest fix information
- Accessing faxes

AIX/PitStop provides access to information that is maintained by RISC System/6000 hardware engineers and AIX software engineers, which keeps you up-to-date on the latest information. More AIX/PitStop information is available on the Internet (See URL in Internet References at the end of this article).

### FixDist

FixDist, another AIX service tool, makes it easy to download AIX fixes and their prerequisites from an anonymous ftp server via the Internet. Since FixDist is updated nightly, it provides a quick, easy way to keep your system current. If you do not want to download the fixes immediately, specify a later time for the download. Check all the specifics on the Internet. (See URL in Internet References at the end of this article.)

## FTP-based Services

Three ftp-based services are available:

**Fix Distribution:** FixDist, a free AIX service tool, makes getting fixes much easier. Use FixDist to download AIX files and their prerequisites from an anonymous ftp server via the Internet. For more information about FixDist, get the file ftp://service.software.ibm.com/aix/tools/fixdist/README.

**Emergency Fix Service:** Another ftp site houses emergency fixes, which are provided as short-term fixes before the formal PTF is available. Since these fixes are provided "as is", you will typically not use this service unless you are working directly with IBM on a problem. If you are directed by IBM, go to ftp://software.watson.ibm.com/pub.

**Data Exchange Service:** If you live in the U.S., use this service to upload your test case or related materials (instead of mailing them to us): ftp://testcase.boulder.ibm.com.

## E-mail-based Services

There are two E-mail-based services: Program Services which provides free defect reporting, and Mail List Server.

**Program Services:** If you do not have an AIX Support Family contract, report suspected software defects by sending E-mail to aixsupt@service.software.ibm.com. If you are outside the U.S., check with your country representative. Each country has its own software defect reporting path.

Since Program Services provides limited support, read your IBM product license agreement and service announcements for more details.

**Mail List Server:** Worldwide customers can subscribe to our AIX Mail List Server to receive E-mail on subjects such as security notifications and other support-related notices. Send mail with a subject of "help" to aixserv@austin.ibm.com for more information.

| Home Page | URL |
|---|---|
| AIX Support Network | http://service.software.ibm.com/www/support/aix |
| FixDist | http://service.software.ibm.com/pbin-usa/fixdist.pl |
| IBM | http://www.ibm.com |
| IBM Global Network | http://www.ibm.com.globalnetwork |
| IBM RISC System/6000 | http://www.austin.ibm.com |
| AIX/PitStop | http://service.software.ibm.com/www/support/aix/pitstop/index.html |

**Figure 2. Internet addresses for support**

## Internet References

Figure 2 provides a list of useful Internet addresses for information and support.

## Summary

The service tools, ftp-based and E-mail-based services, and the fee-based AIX Family of services can help you handle nearly any problem or emergency. The AIX Welcome Packet, AIX/PitStop, FixDist, and access to the Internet provide a wealth of knowledge at your fingertips.

Using these resources will help you understand what services are available and how AIX Support Line works.

**Georgia A. Gibson**, IBM Corporation, 11400 Burnet Road, Austin, TX 78758. Internet: ggibson@ausvm1.vnet.ibm.com. A graduate of the University of Texas in Austin, Ms. Gibson is an AIX service planner in the AIX Service Development group within the RISC System/6000 Division. Her experience includes relationship management, hardware and software usability, as well as user interface design for the SMIT, Distributed SMIT, and VSM for AIX.


## Barriers to Online Commerce Diminish

MasterCard International and Visa International, with support from IBM and other technology partners, have agreed on a standard for doing business securely over the Internet and other public networks. This agreement means that people who would like to buy or sell goods and services on the Internet, but have been concerned about the security of their credit card numbers and other private information, will soon be able to conduct their business without worry.

The new standard, called Secure Electronic Transactions (SET), is the result of a collaborative effort in which IBM contributed technological expertise and a technology called iKP, developed at IBM Research. The iKP technology uses a mathematical technique called cryptography to enable secure electronic payments on the Internet.

SET was developed in accordance with principles that create equal opportunity for thousands of companies and individuals who have innovative ideas for doing business online. For example, in addition to ensuring a high level of security for electronic commerce, the SET protocol is open and available to software developers free of charge, will be fully published, is free of royalty payments, and incorporates technologies that are already standard to the industry.

The next step is for companies to develop and market offers that incorporate the SET protocol.

# Multi-user Solutions: ISA 8 and ISA 128 Multiport Adapters


**By Eddie Ho, Derwin Gavin, Walter Lipp, and Dan Ayala**

AIX is a multi-user system that allows many users to access the system resources. Users can be connected to the system either locally via a direct attachment ASCII device or through network access using a LAN device with TCP/IP protocol. In a host-centric environment in which many active users are draining CPU resources, the ISA 8- and ISA 128-port asynchronous adapters can off-load some character processing from the system.

The RISC System/6000 (RS/6000) has redefined the commercial data processing landscape with its many open system features, including portability, availability, scalability, and ease-of-use. The solution portfolio is mature and well-accepted in the market. This defines the conclusion of phase one of the RS/6000 technology.

Phase two generally includes vertical and horizontal system-level expansions, such as Symmetric Multiprocessing (SMP) and Massive Parallel Processing (MPP) technology, and adoption of industry-standard architecture, such as ISA and PCI. The ISA 8- and ISA 128-multiport adapters complement the industry-standard support. Both adapters are supported as part of the AIX Version 4 TTY subsystem infrastructure on systems that have an ISA bus.

The TTY subsystem has been redesigned to use the STREAMS layered architecture, which provides flexibility for future expansion and can react to more system-level modifications. (See "TTY Subsystem in SMP," *AIXpert*, February 1995.)

## ISA 8-Port Adapter

This multi-channel, intelligent serial communications adapter contains 128 Kbits of dual-ported high-speed RAM for program code and buffering. Each RS-232 port supports speeds of 115 Kbits per second.

The dual-ported RAM, accessible as Read/Write from both AIX and the adapter logic, facilitates high-speed data exchange between the TTY subsystem and the adapter. Figure 1 shows the adapter and the specification.

### Installation

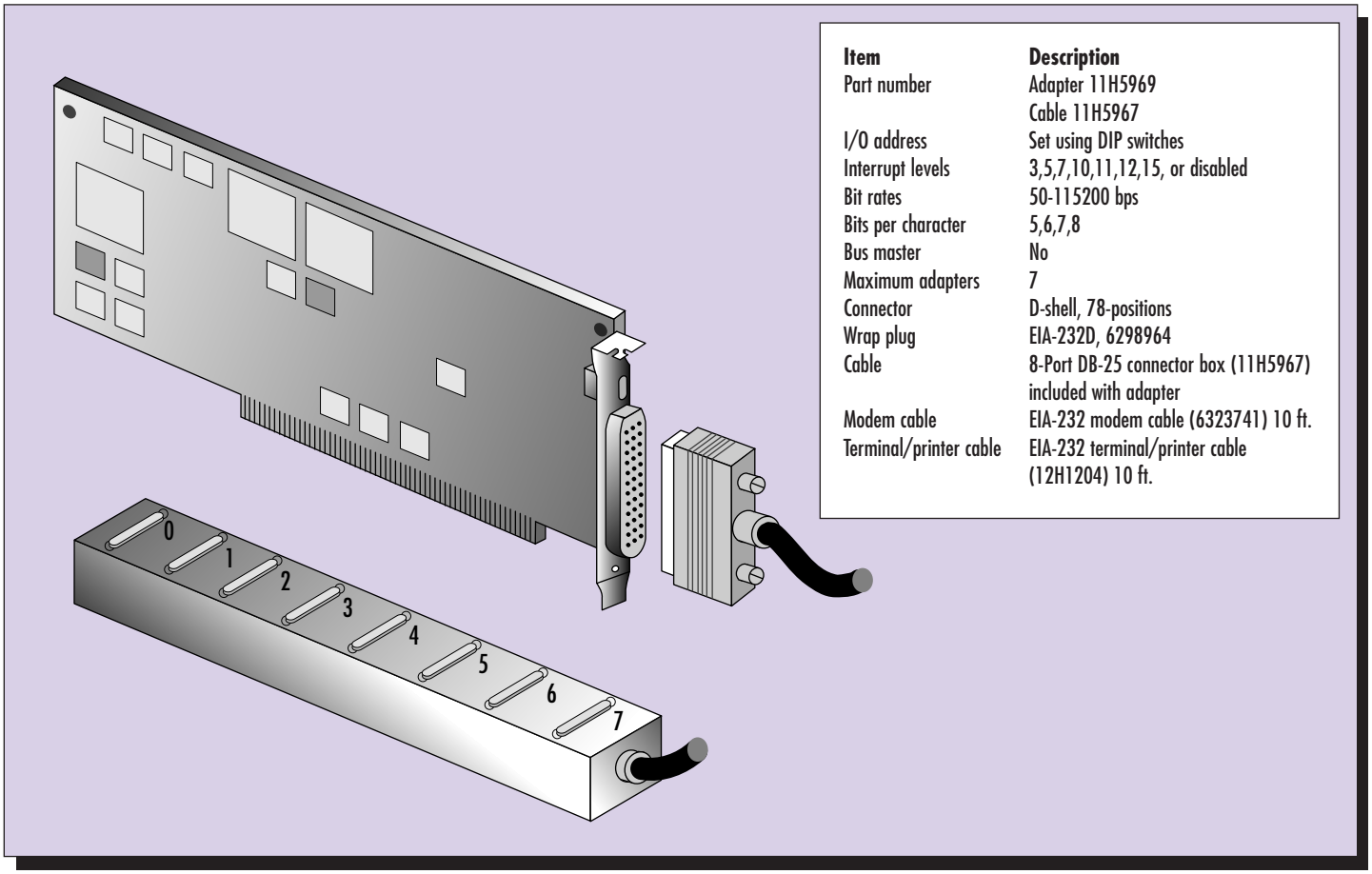
Bus I/O address assignment is a key part of ISA adapter installation. Each ISA adapter within a system must have a unique address assignment. Multiple 8-port adapters can be installed if the address assignments are unique and ISA slots are available.

The default address from the factory is 0x324. If other ISA adapters are already installed, the `lsresource` command (see section titled "AIX `lsresource` Command") can help you determine what address has been selected. The administrator should record this address for later use in software configuration.

Figure 2 shows the address and switch setting.

After the adapter has been properly installed, the next step is device driver installation, which is required for both new or existing systems. In AIX, the Base Operating System (BOS) level custom installation loads all the MCA adapters, but will not load any ISA adapters. The `devices.isa.cxia` driver supports this. Use the AIX `smit isa` command to install this fileset.

## ISA 8-Port Adapter




| Item | Description |
|---|---|
| Part number | Adapter 11H5969<br>Cable 11H5967 |
| I/O address | Set using DIP switches |
| Interrupt levels | 3,5,7,10,11,12,15, or disabled |
| Bit rates | 50-115200 bps |
| Bits per character | 5,6,7,8 |
| Bus master | No |
| Maximum adapters | 7 |
| Connector | D-shell, 78-positions |
| Wrap plug | EIA-232D, 6298964 |
| Cable | 8-Port DB-25 connector box (11H5967) included with adapter |
| Modem cable | EIA-232 modem cable (6323741) 10 ft. |
| Terminal/printer cable | EIA-232 terminal/printer cable (12H1204) 10 ft. |

**Figure 1. ISA 8-port adapter and DB-25 connector**

## Address and Switch Settings

| Bus I/O Address | Address Switch | | | |
|---|---|---|---|---|
| | 1 | 2 | 3 | 4 |
| 0x104 | OFF | OFF | ON | ON |
| 0x114 | OFF | ON | OFF | ON |
| 0x124 | OFF | ON | ON | ON |
| 0x204 | ON | OFF | OFF | ON |
| 0x224 | ON | OFF | ON | ON |
| 0x304 | ON | ON | OFF | ON |
| 0x324 | ON | ON | ON | ON |

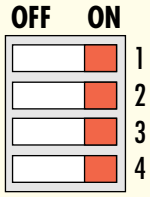



**Figure 2. Address and 8-port switch setting**

### Configuration

After the adapter has been installed, the adapter and the device type of each port must be configured to AIX. Follow these steps to configure the adapter.

1. Enter `smit isa` from the AIX prompt.
2. Select 'Add an ISA Adapter' from the panel. Figure 3 shows the list of ISA drivers that are installed.
3. Select 'pcxr' for bus architecture, as shown in Figure 4.
4. Select the appropriate bus and press Enter. Figure 5 shows the configuration parameters for the adapter.
5. Enter the recorded bus I/O address to the bus I/O address field.

Verify the installation using the `lsdev -Cs isa` command, which lists the configured ISA devices. The 8-port asynchronous ISA adapter should be listed and in the Available state.

The next step is defining each device type within the adapter. There are many parameters in a device definition, but they can generally be categorized as related to transparent printing, DB-25 signaling control, or data characteristics and speed.

This definition allows terminals, printers, plotters, modems, or other special devices to be attached. Refer to the *AIX Version 3.2 and Version 4.1 Asynchronous Communication Guide* (SC23-2488) for all session parameter descriptions. This is the first area for problem determination if the device is not working properly.

## ISA 128 Ports

The ISA 128-Port adapter subsystem can support up to 128 devices offering similar capabilities to the 8-port version. This adapter, also available in Micro Channel® format, is an intelligent adapter with the ability to off-load line protocol processing from AIX. The subsystem consists of an adapter card with up to eight Remote Asynchronous Node (RAN) units, each supporting 16 devices. The adapter is connected to the remote asynchronous node using either a direct connection or a synchronous modem to a remote location. The wiring system allows a remote workgroup in a host-centric application environment to be connected efficiently.

Figure 6 shows a typical headquarters data center with a distributed regional center and a sister center with affinity to the regional center. The corporate headquarters usually consists of multiple applications or data sources on multiple platforms. The RS/6000 can be used as a regional gateway for applications and data access. The sister center is usually tied in for workgroup access due to geographic affinity.

**Adding an ISA Adapter**

```
                    Add an ISA Adapter

Move the cursor to the desired item and press Enter.

tokenring  isa   IBM 16/4 PowerPC Token-Ring Adapter (ISA)
ethernet   isa   IBM ISA Ethernet Adapter
articmpx   isa   N/A
cxia       isa   IBM 128-Port Async, EIA-232 (ISA)
pcxr       isa   IBM 8-Port Async, EIA-232 (ISA)


F1=Help          F2=Refresh       F3=Cancel
F5=Image         F10=Exit         Enter=Do
/=Find           n=Find Next
```

**Figure 3. Adding an ISA adapter**

**Parent Device**

```
                    Parent Device

Move the cursor to the desired item and press Enter.

bus1             Available 00-10 ISA Bus


F1=Help          F2=Refresh       F3=Cancel
F5=Image         F10=Exit         Enter=Do
/=Find           n=Find Next
```

**Figure 4. Selection for bus architecture**

**Adding an 8-port Asynchronous Adapter**

```
        Add an IBM 8-Port, EIA-232 ISA Adapter

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                [ Entry Fields ]
Device Class                    adapter
Device Subclass                 isa
Device Type                     pcxr
Parent Device                   bus1
Bus I/O Address                 0x324


F1=Help    F2=Refresh   F3=Cancel   F4=List
F5=Undo    F6=Command   F7=Edit     F8=Image
F9=Shell   F10=Exit     Enter=Do
```

**Figure 5. Adding an 8-port asynchronous adapter**

### Installation and Configuration

The installation and configuration procedure is similiar to the ISA 8-Port adapter except the following:

1. Bus I/O address (Figure 7 shows the switch diagram for the 128-port adapter.)
2. Software driver fileset is `devices.isa.cxia128`
3. Adapter name is `cxia#`
4. Different bus I/O address (default address is 0X328)

## AIX lsresource Command

The steps below can help determine what bus I/O address has been used by the existing adapter.

**Distributed Environment Solution**

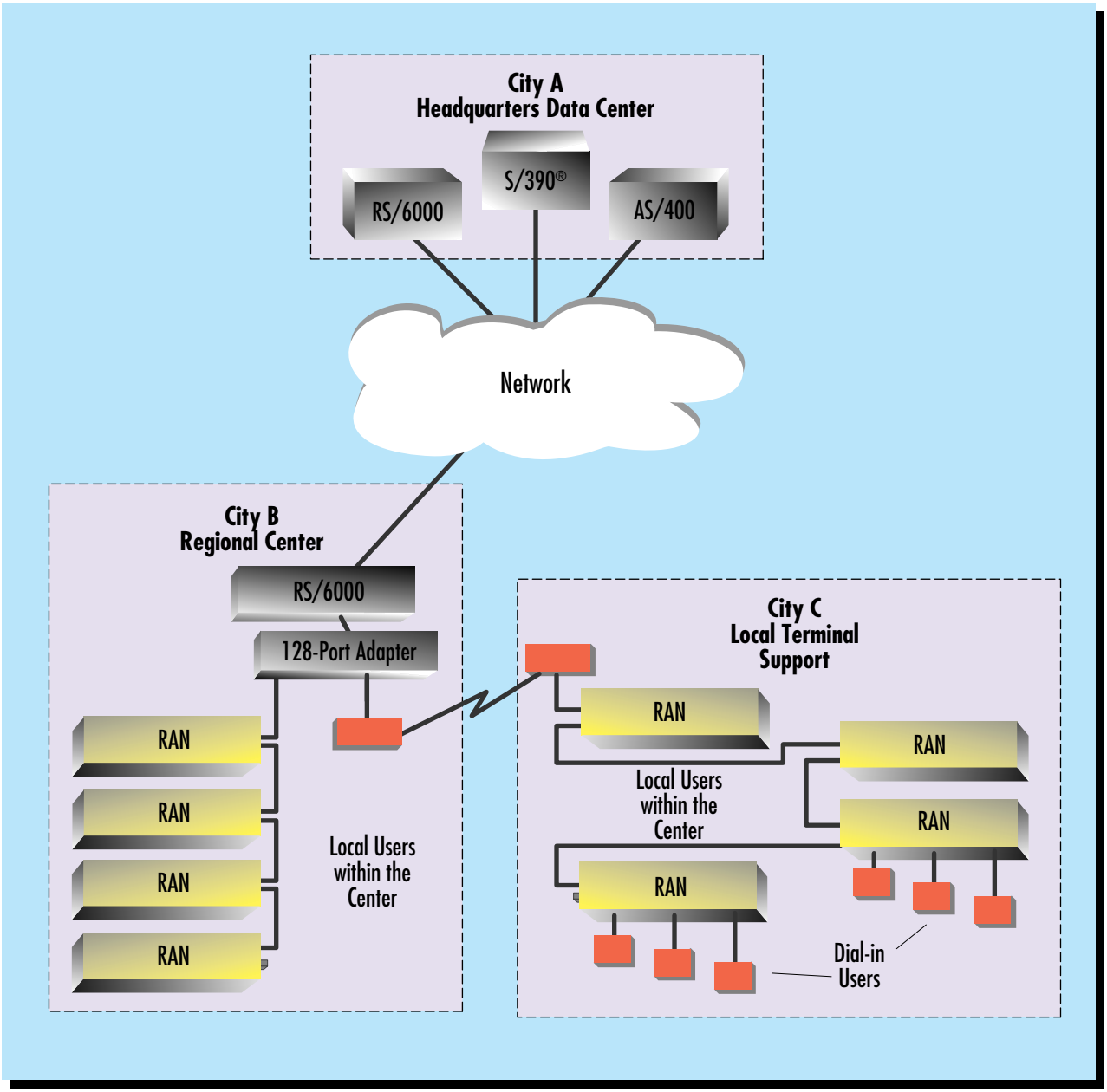



**Figure 6. Distributed environment solution**

1. Enter the following:

```
lsresource -l bus0 -a ] grep bus_io.
```

This command generates a list of adapters using bus I/O that is similiar to the chart in Figure 8.

The candidate bus I/O addresses for both ISA adapters are as follows:

- **8 Port:** 0x104, 0x114, 0x124, 0x204, 0x214, 0x304, 0x324
- **128 Port:** 0x108, 0x118, 0x128, 0x208, 0x218, 0x308, 0x328

The example shows that 0x2a0 is used by ampx0 and 0x6a0 by apm0; therefore, the selected address for 8 Port is 0x324 and 128 Port is 0x328.

2. Set the DIP switch accordingly.

## Conclusion

The ISA 8- and ISA 128-port adapters complete the multi-user solution in a small workgroup environment. As the workgroup increases in size and users demand better scalability, the future solution will be a new generation of multi-user adapters using higher bandwidth PCI bus architecture.

**Eddie Ho**, IBM Corporation, 11400 Burnet Road, Austin, TX 78758. Mr. Ho is a senior programmer in the AIX Executive Briefing Center. He has a BS in Computer Science from the University of Wisconsin and an MS in Computer Science from North Dakota State University.

**Derwin Gavin**, IBM Corporation, 11400 Burnet Road, Austin, TX 78758. Mr. Gavin is a senior associate programmer in AIX Communications Development. He has a BS in Computer Science from Grambling State University in Louisiana.

**Walter Lipp**, IBM Corporation, 11400 Burnet Road, Austin, TX 78758. Mr. Lipp is a senior associate programmer in AIX Communications Development. He has a BS in Electronics Engineering Technology from DeVry Institute of Technology in Chicago.

**Dan Ayala**, IBM Corporation, 11400 Burnet Road, Austin, TX 78758. Mr. Ayala is a programmer in AIX Communications Development. He attended the University of Texas at Austin.

**Switch Diagram for 128-Port Adapter**

| Bus I/O Address | Address Switch | | | |
|---|---|---|---|---|
| | 1 | 2 | 3 | 4 |
| 0x108 | OFF | OFF | ON | ON |
| 0x118 | OFF | ON | OFF | ON |
| 0x128 | OFF | ON | ON | ON |
| 0x208 | ON | OFF | OFF | ON |
| 0x228 | ON | OFF | ON | ON |
| 0x308 | ON | ON | OFF | ON |
| 0x328 | ON | ON | ON | ON |

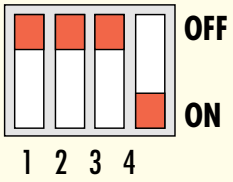
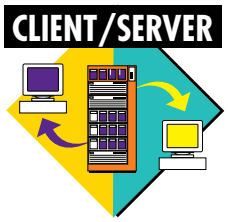



**Figure 7. Switch diagram for 128-port adapter**

| Type | Adapter | Attribute | S G | Current Value |
|---|---|---|---|---|
| o | fda0 | bus_io_addr | | 0x000003f0 - 0x000003f5 |
| o | fda0 | bus_io_addr2 | | 0x000003f7 |
| o | sa0 | bus_io_addr | | 0x000003f8 - 0x000003ff |
| o | sa1 | bus_io_addr | | 0x000002f8 - 0x000002ff |
| o | sioka0 | bus_io_addr | 1 | 0x00000060 |
| o | sioma0 | bus_io_addr | 1 | 0x00000060 |
| o | ppa0 | bus_io_addr | | 0x000003bc - 0x000003be |
| o | ampx0 | bus_io_addr | | 0x000002a0 - 0x000002a6 |
| o | apm0 | bus_io_addr | | 0x000006a0 - 0x000006a6 |
| o | scsi0 | bus_io_addr | | 0x0000f000- 0x0000f0ff |
| o | scsi1 | bus_io_addr | | 0x01000000 - 0x010000ff |

**Figure 8. List of sample adapters with their unique bus I/O addresses**




# Client/Server Success at IBM

By Eddie Ho and Peter Stoll

The Advanced Workstation Systems Integrated Manufacturing (AWSIM) project is a successful client/server implementation in the industrial environment that uses the RISC System/6000 (RS/6000) to run the IBM Austin manufacturing (MRP) process in a high-availability environment. The AWSIM system is the first large-scale industrial application in IBM manufacturing today that runs on AIX.

In 1994, the MIS team at IBM's RISC System/6000 manufacturing plant in Austin, Texas made the decision to re-engineer the information technology that supported the production and inventory control process. This decision was significant because it represented a commitment to the open systems computing model. It was also a chance to use the RS/6000 in both a client and server role. The decision resulted in Advanced Workstation Systems Integrated Manufacturing (AWSIM), the first large-scale industrial application in IBM manufacturing today that runs on AIX.

The importance of this decision to deploy a client/server computing configuration in an industrial environment can be better understood when viewed against the backdrop of the computer industry in general. The decade of the '90s has brought enormous changes in information technology. The mainframe or host-centric computing model of the previous three decades has given way to a whole new technology called client/server or network-centric computing. This represents a fundamental paradigm shift from the concept of a computer manufacturer designing and building proprietary products that required the customer to become dependent on a single vendor for all hardware, software, and services.

The client/server model is based on the concept of world-wide design standards for hardware and software vendors. The widespread adoption of a standards-based approach to information technology has created the open systems or heterogeneous computing environment. For the first time in the history of the information age, customers can control their computer architecture, select products from various vendors, and expect them to interact together seamlessly.

IBM Austin was the logical place to spearhead the use of client/server technology throughout the various processes that govern the development and manufacture of the RS/6000. IBM Austin uses the RS/6000 in several processes including chip simulation, computer-aided design, and software development and testing. In manufacturing, it uses the RS/6000 in processes such as shop-floor control, preloading of software, and systems testing. It is within this context of a site-wide re-engineering of information technology that IBM Austin decided to convert from a mainframe-based Materials Resource Planning II (MRP II) to a client/server architecture.

## Project Scope

The Austin MIS team designed a manufacturing information strategy for the open systems marketplace. This strategy consisted of four objectives. The first was to enable the RS/6000 factory to support the concept of mass customization (shown in Figure 1). Mass customization is the ability to fulfill individual customer sales configurations in high volume. The marketplace now expects an expanding base of options when purchasing computers either one at a time or in large quantities. The Austin team determined the need for new manufacturing (MRP) software to support re-engineering the RS/6000 customer fulfillment process.

Figure 2 shows how a sales configuration is converted into a manufacturing configuration, which in turn draws inventory through the supply pipeline. With this technique, the investment in inventory is tied directly to customer orders, not to a forecast.

The second objective was a just-in-time approach to regulate the flow of inventory within the manufacturing process (Figure 3) as well as the total supply chain from suppliers and trading partners to retail outlets and dealers (Figure 4). The intent is to move inventory only when there is a demand for it further up the inventory pipeline. This just-in-time concept is also described by a Japanese term *kanban*, a signaling system control inventory flow. This approach minimizes the investment tied up in parts on the floor or in finished goods. The Austin team considered the need for cascading replenishment signals that flowed from finished goods backward as an integral part of the new manufacturing software.

The third objective was concurrent engineering (Figure 5), the ability to schedule more development and manufacturing tasks to occur in parallel order instead of in sequential order. The whole RS/6000 Division is continuously shrinking the overall cycle time to market for its new products by organizing many process steps that overlap and by deploying information technology that enables workers with instant access to data and to each other. The manufacturing team considered the need for relational database technology for easy access to information and for flexible software that could be quickly and easily modified when the business process demanded it.

The fourth and final strategic objective was to develop solutions for the first three objectives within the framework of an integrated client/server architecture operating in a network-centric environment. The manufacturing team narrowed their search for hardware, software, and NetWare® solutions that were compatible with the open systems computing model to the RS/6000 products that were designed and built in Austin.

## The Business Perspective

The AWSIM system is a totally integrated manufacturing solution for logistics control and customer fulfillment. The AWSIM system has enabled the factory to attain its first key challenge: build-to-order processing. The system helped the factory open new marketing channels by providing a robust sales order-entry and product configuration module that is tied directly to the shop floor control module. This new function allows the Austin facility to compete in the fast-growing computer catalogue telemarketing industry.

The sales order creates a unique configuration of a base computer model with optional memory upgrades, special cards, preloaded software, and peripheral gear such as keyboards, monitors, and CD-ROMs. This configuration is converted into an umbrella shop order with

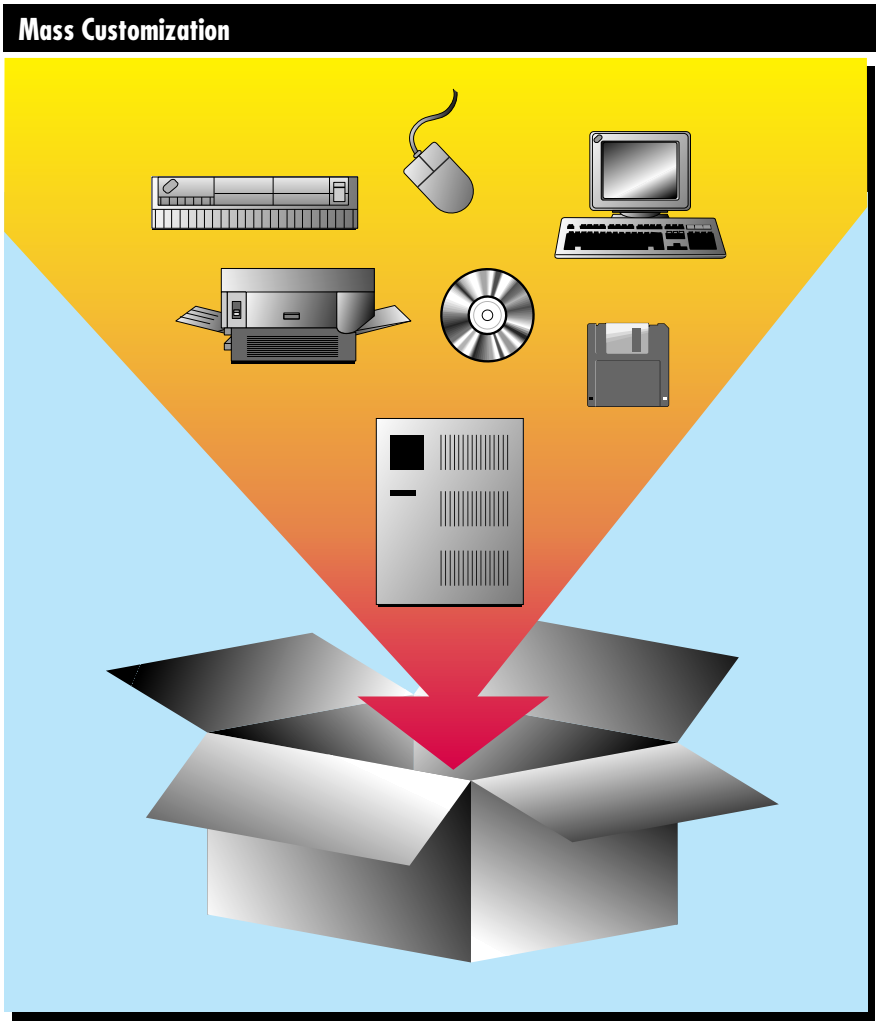



**Figure 1. Customized configurations built to order—options, choices, selections**

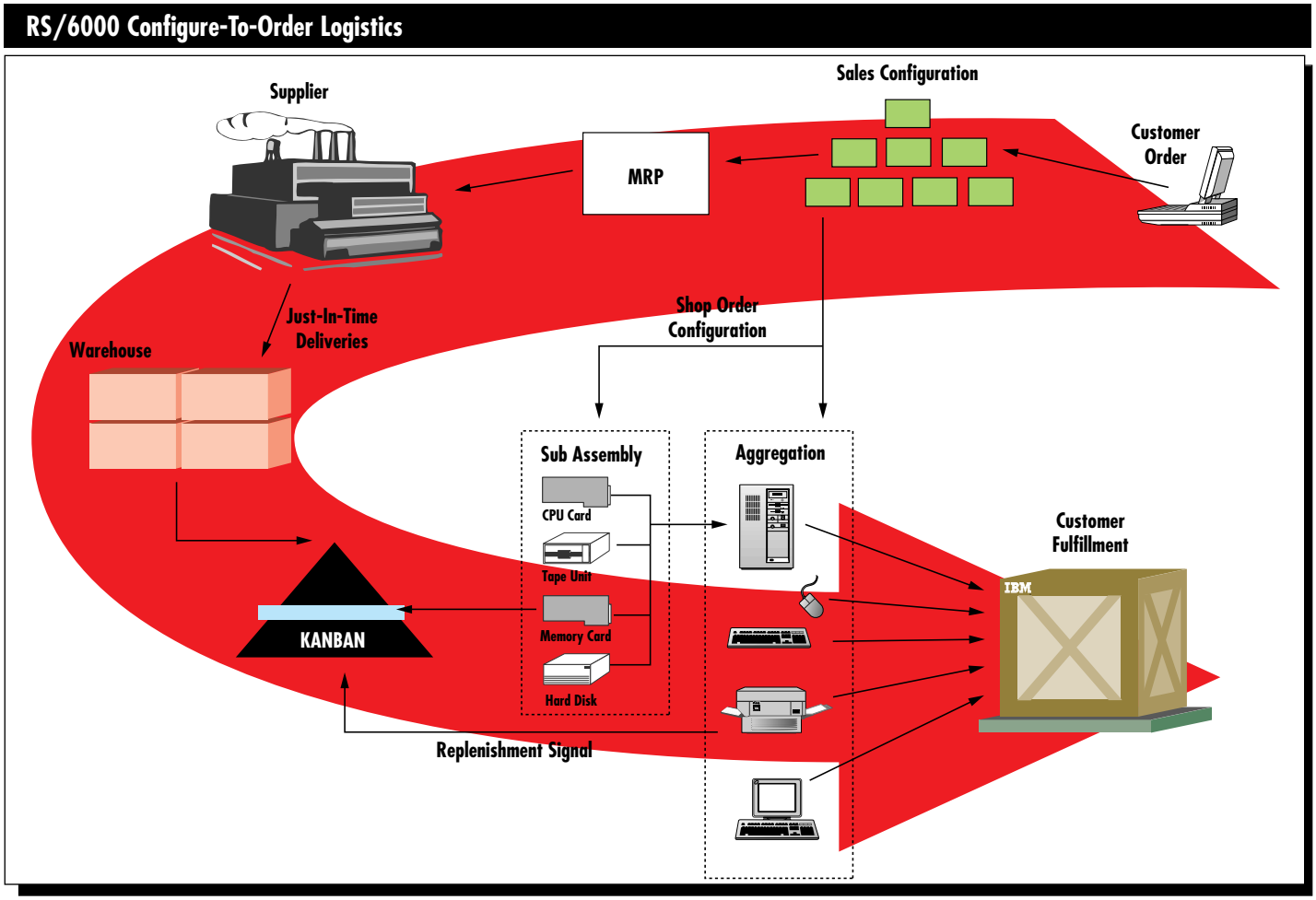



**Figure 2. Converting a customer order into a manufacturing configuration**

nested shop orders for subassemblies, components, and accessory kits. The AWSIM software is an example of how state-of-the-art information technology can facilitate the business goal of reducing the overall fulfillment cycle—from receiving the order to shipping to the customer.

The second key challenge for the AWSIM team was to implement a just-in-time strategy for inventory control throughout the factory and to the suppliers. The key design point was the concept of replenishment triggers that fire whenever the inventory buffer or kanban is depleted at the manufacturing or final packaging workstations. As the operator installs a subassembly (hard file or tape drive) into the computer chassis, a bar-code reader sends a transaction to the server to fire the replenishment trigger to the warehouse. This restocks the line based on the minimum/maximum rules governing that individual inventory item. The Warehouse Management module receives the inventory "pick" request and automatically determines the specific bin location in the warehouse from which to pick the parts. Whenever the warehouse inventory falls below a predetermined minimum level, the AWSIM system automatically sends an electronic replenishment message to the appropriate supplier.

The Austin MIS team has successfully implemented a client/server solution that represents the cornerstone of their vision for the factory of the future. The team developed a statement of requirements based on management's direction to move the plant toward a customer-based fulfillment process. The AWSIM system supports the build-to-order strategy that is quickly becoming the industry norm for computer manufacturing. The ability to configure an order, create a real-time quote, calculate a promised delivery date, and do an instant credit check—all within a five minute phone call—are just a few of the

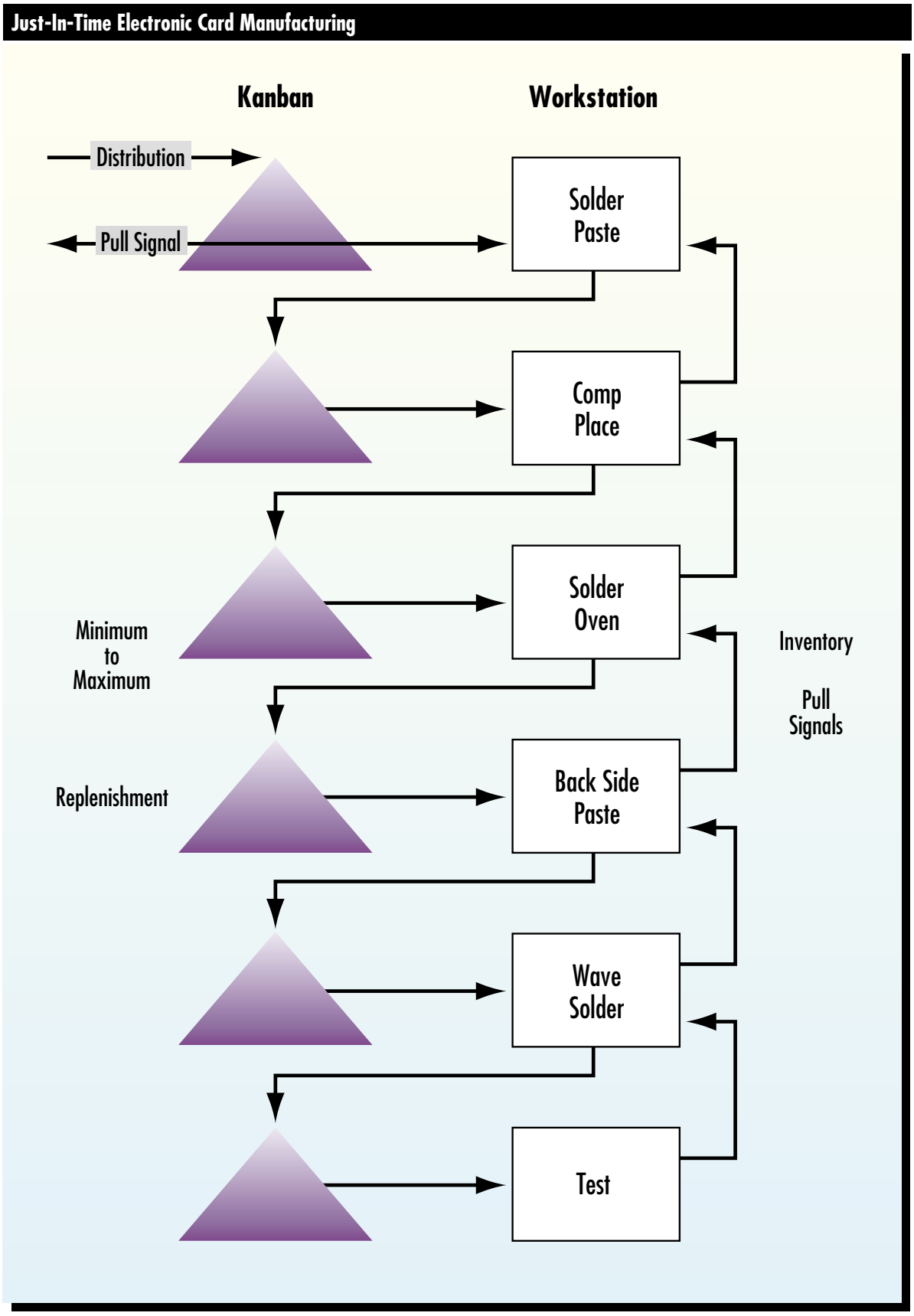



**Figure 3. Regulating inventory flow within manufacturing**

specialized requirements needed by the modern computer factory to compete in the competitive computer catalogue market.

## The Technical Perspective

The AWSIM system is a hybrid version of CIIM 9.0 from Avalon Software Company. AWSIM has been modified to meet the unique demands of the Austin RS/6000 manufacturing and inventory control process. The team recognized that the business needed flexible software that was easy to change when the business demanded it; the older software is generally more rigid and more costly to modify. Avalon Software is written in C using SQL, the standards-based language used by relational database products. This technology decision became one of the critical success factors for the project because new function could be quickly developed on top of the base software package as new requirements arose.

The choice of database products was another key consideration. The open systems marketplace has many excellent database products. The key requirement for Austin was a relational database that performed in a high volume, online transaction environment and delivered subsecond response time.

The team chose the Sybase® relational database management system, which uses performance optimizing techniques intended to focus most real-time processing inside the database, thus minimizing network traffic. The main concepts are stored procedures and triggers. Stored procedures are mini-programs that are preparsed and precompiled to run with minimum overhead from the operating system. The stored procedures are connected by triggers that fire in a predetermined sequence based on the individual transaction.

Client/server technology is designed around the concept of "request and response." The client computer is attached to the network and sends requests for information and/or action to an array of servers also attached to the network. The server is an individual repository of specialized information that sends back a tailored response to the client's request. Client/server is a version of the distributed computing environment that has moved the information age to a new plateau. With this technology, the end user equipped with a workstation can access information across the network from multiple servers and synthesize the data into knowledge at the desktop.

The client and server machines are attached to a 16 Mbit Token-Ring LAN. The whole IBM Austin campus is being retrofitted with 16 MB LANs attached to a 100 MB fiber-optic backbone. The site has deployed router and hub technology for network connectivity and has standardized on TCP/IP as the network protocol.

The hardware solution for the AWSIM database server is a pair of Model 990 RS/6000s. The primary server is backed up by a secondary server in case the primary machine goes down. The software that keeps the two servers operating in tandem is High Availability Clustered Multiprocessing (HACMP). The feature controls the interaction of the two servers and automatically transfers control to the backup when called upon to do so.

In addition to the redundant hardware strategy, the AWSIM system uses Sybase's Replication feature that controls the data mirroring process between the primary and secondary servers. Every transaction that updates

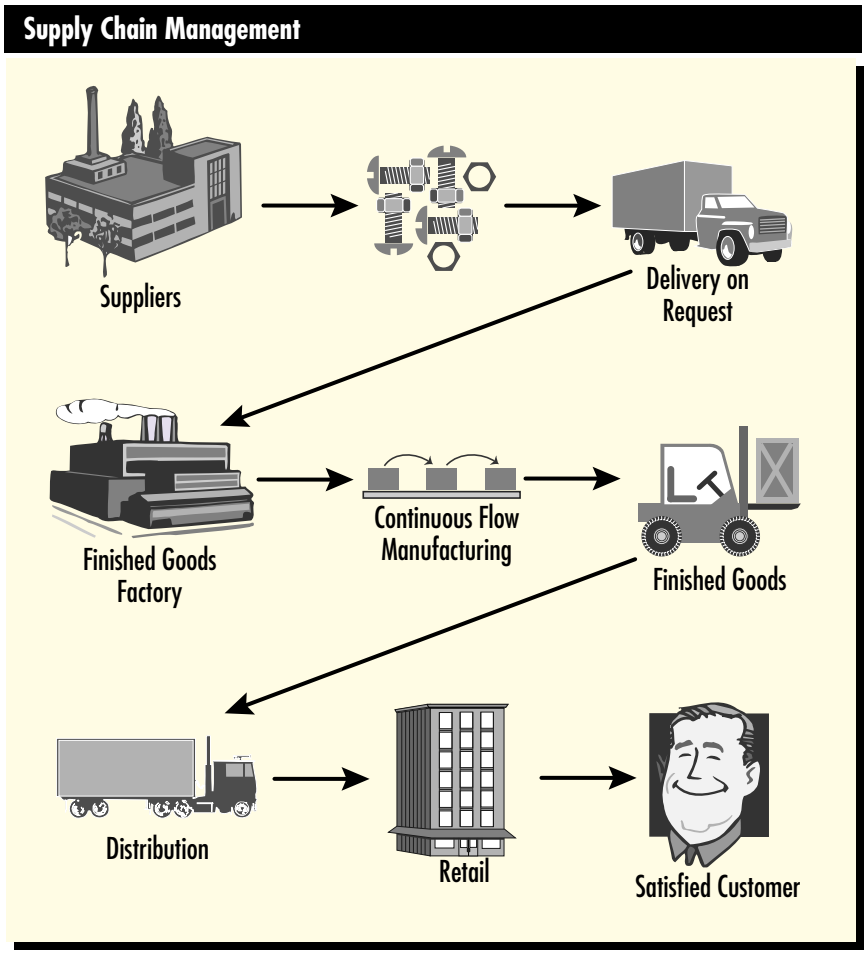



**Figure 4. Supply chain management**

the primary database is copied to the secondary in order to maintain data redundancy between the two environments. The secondary server, which is also used for ad hoc queries by the end users, helps to reduce the load on the production machine.

## Critical Success Factors

The two key challenges to the success of the AWSIM project were scope and skills. The critical factor that contributed most to the successful implementation of the AWSIM system was the team's ability to stay focused on the business requirements and to draw together all the appropriate skills required to support the business and technical considerations.

The AWSIM project required two distinct skill groups. The first group was the business process analysts who mapped out those transactions needed by the user community to do their jobs. For example, the procurement buyer must be able to process a quote from a series of vendors and place a purchase order for the vendor with the lowest price and highest quality. The AWSIM system needed the corresponding transactions to support the buyer's business process. The business analysts wrote the programming specification for the AWSIM development team to follow.

The technical team was the second skill group. This team was further subdivided into internal program designers, development programmers, and system administrators. The nature of client/server or open systems technology is the interoperability of many products within a distributed environment. Accordingly, the AWSIM project required a variety of diverse and specialized computer skills to assist in managing.

The internal design engineering and development programming was done primarily by specially trained staff from third-party software vendors skilled in the native fourth-generation language of Sybase. Their programming proficiency significantly contributed to the rapid rollout of new code whenever the need for functional enhancement arose. An IBM product, Configuration Management and Version Control (CMVC) was the software library and change control tool used to manage the many versions of the software as it progressed through the development and test phases of the project.

The other AIX, security, database, network, and test skills were assembled from various organizations within Austin. The infrastructure team was responsible for keeping the system up all day, ensuring speedy response time, and managing a systems environment that changed continuously.


**Figure 5. Concurrent engineering**

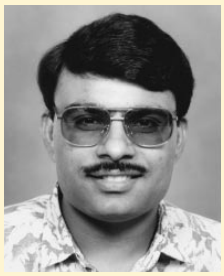
**Eddie Ho**, IBM Corporation, 11400 Burnet Road, Austin, TX 78758. Mr. Ho is a senior programmer in the AIX Executive Briefing Center. He has a BS in Computer Science from the University of Wisconsin and an MS in Computer Science from North Dakota State University.

**Peter Stoll**, IBM Corporation, 11400 Burnet Road, Austin, TX 78758. Mr. Stoll is a senior manufacturing analyst in the RS/6000 Manufacturing area. He has a BA from the University of California at Los Angeles.

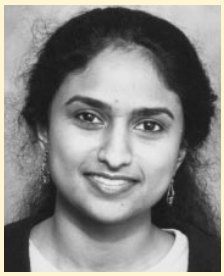

# AIX Questions

**Compiled by Daryl Green**

The AIX Solution Provider Technical Support Group in Austin, Texas, supports software vendors who are developing or porting applications to AIX. This article is a compilation of questions that are frequently asked by vendors. The name of the responding Technical Support Group staff member appears after each response.



Viral Shah



Priya Mvada

### How do I change the default values of `tcp_keepidle` and `tcp_keepintvl` obtained from the `no` command?

Log in as root and run the following commands:

```
/usr/sbin/no -o
  tcp_keepidle=<timeout-value>
/usr/sbin/no -o
  tcp_keepintvl=<timeout-value>
```

Note that `tcp_keepidle` and `tcp_keepintvl` are both measured in half-seconds. For example, `tcp_keepidle=150` is actually 75 seconds. The new settings will remain until you reboot the system. To overwrite default values permanently, include these commands in the `/etc/rc.net` file.

*—Viral Shah*

### Is the procedure for reducing paging space in AIX 4.1.2 different from the standard procedure in AIX 3.2.5?

You can use the AIX 3.2.5 standard procedure to reduce paging space in AIX 4.1.2. The only difference with 4.1.2 is that the primary system dump device is incorporated with the swap device, `/dev/hd6`. So you must first redirect the dump device using the `sysdumpdev -p` command. Then remove the paging space and re-create it with a reduced size by following the AIX 3.2.5 standard procedure.

*—Priya Mvada*

### How do I restore certain specific files from an AIX 4.1.2 `mksysb`? I tried the following, but it did not work:

```
tctl  -f /dev/rmt0 rewind
tctl  -f /dev/rmt0.1 fsf 3
tar xvf /dev/rmt0 ./<path>
```

The `mksysb` in AIX 4.1 is now in backup/restore format; you cannot use `tar` (as you did in AIX 3.2). Use the following command to restore specific files in AIX 4.1.2:

```
restore -so -xvf /dev/rmt0.1 ./<path>
```

*—Priya Mvada*

### Is it possible to install only the `man` pages without having to install the entire InfoExplorer database?

You can select the components that you want from the `/usr/lpp/info/data/ispaths` file and choose to install only those from the InfoExplorer `cd` through System Management Interface Tool (SMIT). If InfoExplorer has already been entirely installed, you can choose the deinstall option from SMIT and remove the components that you do not need. Typically, `man` would require `nav`, `cmds`, `files`, `smith`, and possibly `aixtechref`. You need X11tech and OpenGL™ only if you are doing

related work; man does not search after the seventh stanza, so the rest does not matter.

*—Priya Mvada*

### How can /dev/null be re-created if it was accidently deleted?

Run the following two commands:

```
# rm /dev/null
# mknod /dev/null c 2 2
```

Then change the permissions on /dev/null to 666:

```
# chmod 666 /dev/null
```

*—Priya Mvada*

### How can I obtain a list of kernel extensions currently loaded on the system?

The /usr/bin/genkex command displays the address, size, and pathname for each kernel extension currently loaded on the system.

*—Jeff Simon*

### How can I obtain a list of objects for each process currently running on the system?

The /usr/bin/genld command prints a report containing the process ID and name, followed by a list of objects loaded for that process, as well as each object's address and pathname.

*—Jeff Simon*

### I have Base Operating System (BOS) 4.1.3 installed. How do I increase the number of licensed users on the system?

Do the following as root:

```
<smit/smitty>
   System Environments
     Change / Show Number of Licensed Users
```

Next, reboot the system for any changes to take effect. If you are unable to change this parameter, you may not have the required software installed on your system. Enter the following command to see if this software is installed:

```
lslpp -l | grep bos.sysmgt.login
```

*—Jeff Simon*

### How can I obtain the name of a function being used by a thread within DBX?

Within DBX use the thread command, then look under the function field.

*—Jeff Simon*

### How can I use DBX to display information pertaining to the current thread being used?

This can be obtained with the following:

```
(dbx) thread current
```

*—Jeff Simon*

### How can I set the time/date for a host on a network?

Use the /usr/sbin/setclock command. Token Ring must be installed and TCP/IP must be working to use this command. The setclock command gets the time from a network time server, and sets the local time and date accordingly.

*—Jeff Simon*

### What is the purpose of shr.o file in the C library?

The shr.o file contains system calls and handlers; it is dynamically loadable and sharable by default.

*—Jeff Simon*

### How do I determine whether I have the C for AIX compiler or the C Set++ compiler installed in AIX 4.1?

If both the xlC.C++.cmp and xlC.C filesets are installed, then the C Set++ compiler is installed.

**Jeff Simon**

```
1. cc -qroconst -o <output> <filename>.c  // -qroconst will write to text
2. size -f <output>  –>>

    <output>: 380(.text) + 64(.data) + 16(.bss) + 302(.loader) = 762

3. cc -o <output> <filename>.c              // for comparison
4. size -f <output>  –>>

    <output>: 280(.text) + 164(.data) + 16(.bss) + 302(.loader) = 762
```

**Figure 1. Using -qroconst to compile source code**

If only the `xlC.C` fileset is installed, the C for AIX compiler is installed. (C for AIX is a subset of C Set++ compiler. C for AIX compiles only C programs, whereas C Set++ can compile both C and C++ programs.)

Use `lslpp -l <fileset name>` to check the system's fileset installation.

*—Darshan Patel*

**All of my build tools use the `CC` command to invoke the C++ compiler, but AIX uses the `xlC` command. How do I use `CC` instead of `xlC` to invoke the C++ compiler without modifying build tools?**

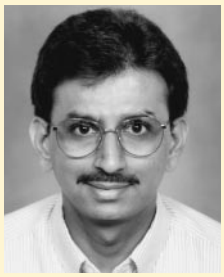
First create a new stanza `CC` in `/etc/xlC.cfg` with the same attributes as `xlC`. Then link `CC` to `xlC`.

*—Darshan Patel*

**I have a stripped object file. When I link the stripped object file, I get an error message "`ld: 0711-710 ERROR: Input file empty is stripped. The file is being ignored.`" Can I link the stripped object file?**

No. Stripping removes (or does not generate) the symbol table. Without a symbol table, the file cannot be linked.

*—Darshan Patel*



**Darshan Patel**

**How can I write to the text segment of an executable?**

Compile the source code with `-qroconst`, as shown in Figure 1.

Compiling with `-qroconst` will write to the text segment (the default is to write to the data segment). Note that pointers and complex aggregates containing pointer members cannot be placed in the text segment.

*—Jeff Simon*

**When I link my program on AIX 3.2, I get the error message "`0706-781 ERROR: TOC OVERFLOW`." What is causing this error message and how can I resolve it ?**

The error message is generated because the program exceeds the maximum TOC size limit of 64 KB.

One way to bypass the problem includes the following:

- Split the program into shared libraries.
- Limit the number of global/static variables.
- Put all the global variables in a single structure and get only one TOC entry for that structure.

Another solution is to install PRPQ P91128, which allows you to exceed the maximum TOC size limit of 64 KB. To obtain this PRPQ, call 1-800-IBM-CALL or contact your local IBM marketing representative. Be sure to specify `-bbigtoc` flag with `ld` to allow for the larger TOC size.

```
usage: dtconfig -e|-d|-kill|-reset
   -e (enable auto-start of dtlogin)
   -enograph (enable auto-start of dtlogin without graphical boot)
   -d (disable auto-start of dtlogin)
   -kill (kill dtlogin)
   -reset (reset dtlogin - reread configuration files)
```

**Figure 2. Desktop configuration options**

The AIX 4.1 binder has removed this limitation by creating second level TOC information.

*—Darshan Patel*

**I am trying to strip a large executable file using the strip command and I am getting error message "0654-404 Cannot write to file". How can I resolve this error message?**

The /tmp filesystem is running out of free space, causing the error message. The strip command uses the /tmp filesystem to create the file and to protect it against being killed. Depending on how the executable was compiled, it may require a lot of storage in /tmp.

If your system is AIX 3.2, you must find more space for /tmp. If your system is AIX 4.1, then you can set the environment variable TMPDIR to a directory to use for the temporary file on a filesystem with more space, and strip will use that instead.

*—Darshan Patel*

**How do I control the behavior of the Common Desktop Environment (CDE) login interface on the system console?**

By default, /etc/inittab sources /etc/rc.dt to start the CDE login by executing /usr/dt/bin/dtlogin when the system boots. This will display a CDE login on your system and to any Xterminals requesting a login via XDMCP. To serve a CDE login to Xterminals only and to have the familiar green and black common character mode displayed on the system console, you must comment out the following from your Xservers file as defined by your Xconfig file:

```
:0  Local local@console
/usr/lpp/X11/defaults/xserverrc -T -
force :0
```

*Note:* By default, the dtlogin resource files are located under /usr/dt/bin and /usr/dt/config. Resource files in these locations may be overwritten when applying system updates; therefore, it is advisable to copy these files to their equivalent paths under /etc . Once these files are copied, further editing may be required to include the location of other resource files.

The desktop configuration utility /usr/dt/bin/dtconfig is used to manage dtlogin. Entering /usr/dt/bin/dtconfig by itself will display the available options as shown in Figure 2.

The -enograph option is similar to the -e option. However, this flag will disable the AIX console message window and the background AIX logo. The system will boot with the messages displayed in common character mode.

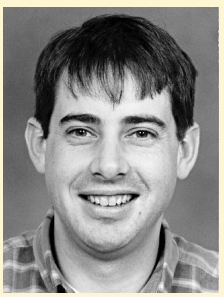
*—David Stewart*

**I have two display adapters (see Figure 3) installed in my system. How do I determine which one is the default, and how do I change it?**

With AIX 3.2.x, you can use # lsdisp.

Next, use the following command to list the attributes of hft0:.

```
lsattr -El hft0 | grep console
```



**David Stewart**

| Name | Description |
|---|---|
| POWER_Gt4xi:1 | Graphics Adapter |
| hiprf3d:2 | High-performance 3-D color graphics processor |

**Figure 3. Display adapters**

You should see the following:

```
console  1  Physical display for HFT
console   True
```

The 1 listed corresponds with the number in the first column from `lsdisp`.

Next, enter the following:.

```
chdev -l hft0 -a default_disp=<adapter>
-a console=<n> -P
```

where adapter is the device listed in the first column of the `lscfg | grep -I graph` output and `n` = the console number. Determine the console number by examining the output of `lsattr -El hft0 | grep console`. If the current console number is 1 and you have two display adapters, then you will use 2 for `<n>`. If the current console number is 2 and you have two adapters, then `<n>` will be 1.

Once the machine is rebooted, this change will take effect.

In AIX 4.1 this operation is much less complicated. You can use the following procedure:

Take note of the `DEV_NAME` column from the output of the `lsdisp` command in Figure 4.

Use this output with the `chdisp` command. The proper usage is shown in Figure 5.

Example: `chdisp -pgda0` will change the colorgda adapter to be the default display adapter upon subsequent reboot.

*—David Stewart*

| DEV_NAME | Slot | Bus | ADPT_NAME | Description |
|---|---|---|---|---|
| ppr0 | 03 | mca | POWER_Gt4x POWER Gt4 | Midrange Graphics Adapter |
| gda0 | 07 | mca | colorgda | Color Graphics Display Adapter |

**Figure 4. Output from lsdisp**

| Usage | Function |
|---|---|
| `chdisp [-dDeviceName] [-pDeviceName]` | Changes the default display |
| `-d` | Changes the default display for this session |
| `-p` | Changes the default display in the database; effective at the next IPL |
| `DeviceName` | Represents the logical name of the display; the name in the first column of output of the `lsdisp` command |

**Figure 5. The chdisp command**

**Sometimes when I bring up a new machine, it hangs on a blue screen with the words "Starting the Common Desktop Environment" or it shows the following error: "The DT Messaging system could not be started."**

Enter the following to correct the problem:

1. Choose [OK] to return to the login screen.
2. Select Failsafe Session from the login screen's option.
3. Make sure the hostname is correct in these locations:

```
/etc/src.sh
/etc/hosts
/usr/adm/inetd.sec
```

The cause of the above scenario is generally faulty network configuration. In AIX 4.1, CDE and the network are very closely related. You can verify the following:

1. Is the hostname set (If you type `hostname`, does it return the proper value)?
2. Can you resolve the hostname of the machine (Is the proper value returned when you type `host <hostname>` or `host <IP-address>`)?
3. Are the network interfaces properly configured (If you type `netstat -in`, do you see the proper interfaces)?

4. Are you using DNS or NIS? Check for the proper configuration (that is, permissions and structure of `/etc/resolv.conf`).

*—Michael Hinegardner*

### How would I prevent users from being able to open a terminal session via the window manager menus in CDE?

Use a "restricted mode," which not only removes the "Open Terminal" option from the window manager menus, but it also keeps users from leaving their home directory when using the file manager (dtfile).

To restrict terminal use, it is necessary to enter the following resource to `/usr/dt/app-defaults/Dtfile`:

```
*restrictMode:  True
```

*Note:* Since this is a resource, it can be placed in any given resource file such as `.Xdefaults`.

*—Michael Hinegardner*

### I am having problems with an application written using the `SIOCGIFCONF ioctl()`.

In AIX 4.1, the implementation of the `SIOCGIFCONF ioctl()` is similar to the BSD Version 4.4. The structures returned are variable length; binary compatibility is provided to AIX 3.2.5 applications. There is also a variant `OSIOCGIFCONF` provided for AIX 4.1 applications that expect the same behavior as AIX 3.2.5. In Version 3.2.5 you must have been setting the `compat_43` option for BSD 4.3 compatibility (`no -a`). AIX 4.1 no longer requires this.

*—Carl Senegal*

### What is the general procedure to follow if the system crashes while booting (flashing 888 on the LED panel)?

Since most crashes are caused by the last event that happened on the system, we can assume that the last LED before the system started to dump is the cause. Note the LED.

- If this LED is configuring some hardware device, contact hardware support for help.
- If this LED is configuring some software, contact your appropriate software support organization for help. Contact your software support organization if you cannot determine what the LED means.

*—Carl Senegal*

### While running an application, I receive an error message saying the maximum number of file locks have been exceeded. How can I tell if there are any unused lock entries?

The maximum number of file locks is limited to 2,000 per process and 200,000 for the system. This is hardcoded and cannot be configured. The `pstat -T` command provides (among other things) the number of lock entries currently available for use.

*—Priya Mvada*

### Is there a way in AIX 3.2.5 to exempt specific users from the password restrictions specified in the `/etc/security/login.cfg` file?

The entry `flags=NOCHECK` can be added to the stanzas of these users in the file called `/etc/security/ passwd`.

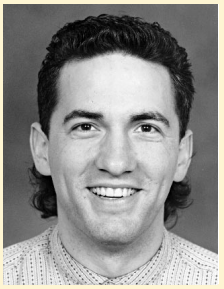
*—Priya Mvada*

### What is the procedure to retrieve just one file (for example, the `/etc/passwd` file) from a backup tape?

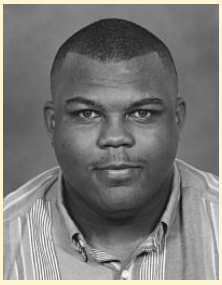
With the backup tape in the drive (which we assume is called `/dev/rmt0`), run the following sequence of commands:

```
tctl rewind
tctl fsf 3
cd /
tar -xvf/dev/rmt0.1 ./etc/passwd
```

*—Priya Mvada*



**Michael Hinegardner**



**Carl Senegal**

**What is the correlation between the return value of the `pthread_self()` subroutine and the thread identifiers assigned by dbx to threads?**

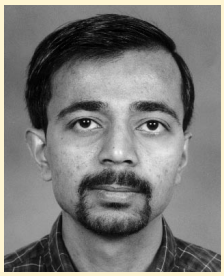
The `thread info` command shows detailed information about each thread. One of the items is labeled `thread (pthread addr):`. The address displayed is eight bytes more than the value returned by `pthread_self()`. The exact number of differing bytes may change from release to release, but the printed address will always be some small number of bytes beyond the `thread_self()` value.

*—Manohara Yarehalli*

**I was porting from AIX 3.2 to 4.1 and I received the following warning: "`0711-327 Warning: entry point not found _nostart`".**

In AIX 4.1, use the `-bnoentry` option instead of `-e _nostart`.

*—Manohara Yarehalli*

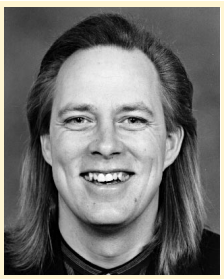


Manohara Yarehalli

**Why does my `lpstat` command print a double entry?**

The reason is because the printer is physically attached to a remote queue. The first entry in the `lpstat` output corresponds to the local queue. The second entry corresponds to the remote queue.

*—Daniel Pedraza*



Craig Stermer

**I installed the Japanese language fileset on my AIX 3.2.5 system, but it is still comes up in English. Why?**

Even though the fileset has been installed, the language environments have not yet been changed. Use the following process to change the language environment:

1. 'Su' to root
2. Run SMIT
3. Select System Environments
4. Select Manage Language Environment
5. Select Change Language Environment
6. Press F4 to generate a list of available languages
7. Select the desired language and press Return
8. Exit SMIT
9. Reboot the system and the change will take effect

*—Daniel Pedraza*

**What is the best way to clear out my old TCP/IP networking interfaces and start over again as if it were a new machine?**

Perform the processes shown in Figure 6.

**Token-Ring**

```
# ifconfig tr0 detach
# rmdev -l tr0 -d
# rmdev -l tok0 -d
```

**Standard Ethernet**

```
# ifconfig en0 detach
# rmdev -l en0 -d
# rmdev -l ent0 -d
```

**802.3 Ethernet**

```
# ifconfig et0 detach
# rmdev -l et0 -d
# rmdev -l ent0 -d
```

**Figure 6. Clear old TCP/IP interfaces**

*—Craig Stermer*

**What is the maximum number of SPX connections allowed with NetWare/6000?**

Version 3.11A allows up to 250 connections.

*—Craig Stermer*

### How many NetWare engines should I run on a server running NetWare/6000?

It is recommended that you run one engine for every 10 users. This is a general recommendation and can change because of system load.

*—Craig Stermer*

### Why do applications using the SO_KEEPALIVE function not die when the application terminates abnormally?

AIX supports applications, such as telnet, that use the SO_KEEPALIVE function and implements the tcp_keepidle and tcp_keepintvl parameters within the no command to keep track of inactivity on established sessions. The supplied default for both AIX Version 3 and Version 4 results in a timeout of 2 hours 10 minutes, which is calculated as follows:

tcp_keepidle = 14400 half seconds or 7200 seconds or 120 minutes or 2 hours

This timer tracks the period of inactivity on a session. If a command is entered, the timer then resets. It waits for two hours before taking any action on an inactive session. When that timer expires, AIX then sends a probe to the original site to determine if the system is still active. If it receives a positive response, it leaves the session in an established state. If it gets no response, it will send a further eight probes (nine total) at an interval designated by the tcp_keepintvl parameter.

tcp_keepintvl = 150 half seconds or 75 seconds

If AIX receives no response after the ninth probe, 2 hours 10 minutes after the session was established, it kills the session and releases all resources associated with the session—if it is able to do so. If an application does not accept the kill signal, the application remains active and the session remains established.

The user can customize this timeout scenario by altering the two parameters in the no command as follows:

```
no -o tcp_keepidle=7200
no -o tcp_keepintvl=75
```

This would cut the default timeout in half if these commands were run from the command line, but they would only be current for the duration of the system boot. When the system is rebooted, these parameters will reset to the system defaults. To overcome this, it is recommended that the user's preferred parameters are set by adding the no -o commands to the end of the /etc/rc.net startup script, which is run at every reboot.

*—Richard Barnett*

### How do I prevent rexec, ftp, rsh, and rlogin from always prompting me for a password?

The rexec and ftp commands make use of a $HOME/.netrc file on the user's local system when connecting to a remote system. This enables the user to provide user account and password details for use at the remote site. The $HOME/.netrc file must have 600 permissions (-rw——-) and be owned by the local user. The format would look similar to the following:

```
machine test1 login user1 password
12345
```

In this case, the local user will be able to rexec or ftp to remote system test1 with user account user1 and password 12345 and would not be prompted for user account or password information. Because the password information is written in plain text, the file must have the correct permissions and ownership as stated above.

The rsh and rlogin commands use a $HOME/.rhosts file on the user's target machine, giving details of which users are allowed to log in from which systems on the network. Again,

the $HOME/.rhosts file should have 600 permissions and be owned by the user account on the remote system. Its format would look similar to the following: test2.austin.ibm.com user2.

In this case, the user user2 would be allowed to login to the remote system from the test1 system without being prompted for user account or password details. Note that if Domain Name Services (DNS) is being used, the fully qualified hostname of the local system is required.

*—Richard Barnett*

### Why is the hostname always reset to aoot.austin.ibm.com every time I reboot?

TCP/IP in AIX Versions 3 and 4 can be set up for BSD style configuration, although the default is not to use this option. If the user uses smit tcpip, selects "Further Configuration" and "Select BSD style rc Configuration", then changes the parameter to yes from the default no, AIX will use the /etc/rc.bsdnet startup script instead of /etc/rc.net by default.

In the /etc/rc.bsdnet file, IBM has provided a sample hostname using the fully qualified hostname aoot.austin.ibm.com. If you do not change this to the required value, then the system will be known as this hostname. You either need to set the "Select BSD style rc Configuration" parameter back to the default no value or change the hostname in /etc/rc.bsdnet.

*—Richard Barnett*



## SDO Moves/Improves WWW Home Page

IBM's Solution Developer Operations (SDO) has restructured its WWW home page to add new functions from the recently announced Solution Developer Program (SDP) that will give information about the new program and allow online registration to become a member of the SDP. During the process, the URL was changed to emphasize the purpose of the page—information for developers.

The new home page includes a section that highlights product announcements from SDP members. These items can also be linked to the SDP member's home page. Complete information about including an announcement in this section is found in the SDP section of the home page.

The SDP section is an area exclusively for SDP members. When you register for the SDP using the online registration process, you receive an ID and password that allows you to access this restricted area of the home page where you find the following:

- Information about each of IBM's Partners in Development programs
- Latest news of interest to solution developers
- An extensive library of reference material and sample code for IBM's platforms
- Information about the facilities available at IBM's Solution Partnership Centers
- A Frequently Asked Questions (FAQ) section for browsing
- Online forums for getting answers to technical questions—for those who have signed up for technical support

Everything you need to get developer support for any or all IBM platforms is available through this one interface. Come visit us at our new URL:

http://www.developer.ibm.com/

and be sure to register for the Solution Developer Program.