AIX Questions

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.

Where can I obtain information about the ANSI C standards?

You can find information about the ANSI C standards from the following URL:

http://www.ansi.org

—Jeff Simon

How can I measure performance of a context switch for threads and processes?

A *context switch* is the switching from one thread to another inside a process. The cost of such an action refers to a performance issue. AIX provides a tool called `vmstat` for monitoring such activities. The number (rate) of kernel-thread context switches can be found under the "cs" column, which is under the "faults" column.

A context switch consists of the following:

- Saving the machine state of the departing process
- Recalling the machine state of the selected process
- Mapping mapping (`u_area`) and other virtual space of the selected process
- Switching CPU to execute with the registers of the selected processes

When another process is given control of the CPU, the context, or working environment, the previous process must be saved and the context of the current process must be loaded. Since AIX has an efficient context switching procedure, each switch is inexpensive in terms of resources. The `vmstat -s` tool can be used to monitor CPU context switching (for example, dispatch of a new process). If there is any significant increase in context switches, you should do some further investigation.

A fileset called `bos.acct` contains `vmstat`. See InfoExplorer™ for additional information about `vmstat`.

—Jeff Simon

How can I determine if a deadlock on AIX has occurred?

Run the `dlock` subcommand to determine if the condition has occurred. A deadlock occurs when two or more threads are waiting indefinitely for an event that can only be caused by one of the waiting threads. When such a state is reached, these processes are considered deadlocked.

—Wade Carlin

Wade Carlin

Jeff Simon

How can I map a file system object into virtual memory using mmap()?

The `mmap()` function explicitly maps a file system object into virtual memory. Figure 1 shows an (unsupported) example.

—Hung Dinh

```
/*    Program shows how to use  function call mmap() to map a
      file system object into virtual memory     */

#include <sys/signal.h>
#include <sys/types.h>
#include <sys/mman.h>

#define SADDR   0x40000000

main()
 {
 int i;
 char *mmptr;

/* mmap function invocation  */

   mmptr = (char *)mmap((char *) SADDR,256 , PROT_READ|PROT_WRITE,
            MAP_ANONYMOUS|MAP_FIXED, -1, 0);

/* SADDR is the starting address of the mem region to be mapped
   256 is the # bytes of the region to be mapped

   PROT_READ   region can be read

   PROT_WRITE  region can be written

   MAP_ANONYMOUS specifies the file descriptor of the file system object
   to be mapped. If the MAP_ANONYMOUS flag is set, the fd parameter
   must be -1. MAP_ANONYMOUS means the creation of a new anonymous mem
   region that is initialized to all zeros

   MAP_FIXED specifies the mapped region be placed exactly at the address
   specified by the addr parm
*/

    if (mmptr == (char *) -1 )
       {
         perror("Problems with mmap ");
         exit(1);
       }

 /* write some data into the mem region to be mapped */

 for(i=0;i<256;i++)
 {
     *(mmptr+i) ='Y';
 }

 /* verify the data is written  */
 /* this should print out 256 'Y's as mapped by the above for loop */
 printf(" mapped data is  %s   \n", (mmptr));
 }
```

Figure 1. An mmap() function0 example

How do I get Common Desktop Environment (CDE) to read my .profile?

The `.dtprofile` file in your home directory controls whether your `.profile` is read. Uncomment (remove the leading #) on the line that contains `DTSOURCEPROFILE=true`.

—Wade Carlin

I want to use mutexes and condition variables across multiple processes, but cannot find the constant PTHREAD_PROCESS_SHARED.

The `pthread_condattr_setpshared()` function is present but not implemented on AIX at this time. It simply returns as shown in Figure 2.

The function `pthread_mutexattr_init()` is also present, but currently not implemented on AIX. The `pthread_mutexattr_t` is the attribute structure, and this function on AIX marks it as being initialized. There is no real initialization to be done because there are no attributes, as shown in Figure 3.

Shared mutexes and condition variables will be added in AIX 4.3. There are no plans to implement this functionality in AIX 4.2 or prior releases.

—David McCloud

What is the performance of different IPCs (semaphores, message queues, pipes)?

Hung Dinh

For any form of Interprocess Communications (IPC), the overhead is usually attributable to the time required to execute the system calls (`read, write, msgsnd`) and the time required to move the data between the processes. Message queues are faster than pipes and FIFOs, since the latter two techniques use the general `read` and `write` system calls. Message queues have their own system calls, which can be implemented efficiently. Since the semaphore calls do not transfer a message between the user process and the kernel, they are the most efficient in terms of speed. The following book details the performance aspect of different IPCs: *UNIX Network Programming* by W. Richard Stevens (ISBN 0-13-949876-1).

—Jeff Simon

I have a huge C++ non-threaded application that I need to port over to use threads; however, I would like to avoid recompiling the application.

For C++ programs, you must recompile the application. This is because applications that use threads must to be compiled with one of the thread-aware compiler invocations

```
int
pthread_condattr_setpshared (const pthread_condattr_t \\*attr,
                             int pshared)
 {
     return (ENOSYS);
 }
```

Figure 2. PTHREAD_PROCESS_SHARED

```
int
pthread_mutexattr_setpshared (const pthread_mutexattr_t    \\*attr,
                              int pshared)
{
      return (ENOSYS);
}
```

Figure 3. The pthread_mutexattr_t attribute structure

(`xlC_r`, `xlC_r4`) to bring in the proper flags (see `/etc/xlC.cfg`; for example, the `options` field, and the order of the libraries listed under `libraries2`). If you did not compile your application using one of the thread-aware compiler invocations, then you must recompile your code.

—Jeff Simon

How do I set up a user with the ability to remotely logon to an AIX machine without being prompted for a password?

The `$HOME/.rhosts` file defines which remote hosts (computers on a network) can invoke certain commands on the local host without supplying a password. See InfoExplorer for the `.rhosts` file format.

—Jeff Simon

What do I look for when the AIX system hangs?

System hangs inside the kernel are generally caused by high-priority (low-value) threads or missed events. A missed event is one that has posted or awakened and now has a thread sleeping on it. The thread sleeping on this event has somehow missed the post or wakeup.

—Wade Carlin

Compiled by Jeff Simon, IBM Corporation, 11400 Burnet Road, Austin, TX 78758.