03/13/96

Discussion of Default Values in /etc/security/limits

Special Notices

Information in this document is correct to the best of our knowledge at the time of this writing. Please send feedback by fax to "AIXServ Information" at (512) 823-4009.

Please use this information with care. IBM will not be responsible for damages of any kind resulting from its use. The use of this information is the sole responsibility of the customer and depends on the customer's ability to evaluate and integrate this information into the customer's operational environment.

Default Values in /etc/security/limits

About This Document

This document applies to AIX versions 3.1 through 4.1.

The following are the default values in /etc/security/limits.

```
        3.2                              4.1
  ====================            ====================
   default:                         default:
       fsize = 2097151                  fsize = 2097151
       core = 2048                      core  = 2048
       cpu = 3600                       cpu   = -1
       data = 131072                    data  = 262144
       rss = 65536                      rss   = 65536
       stack = 8192                     stack = 65536
```

These values are used as default settings when a new user is added to the system. They can be changed when the user is added (mkuser) or after the user is created (chuser). At login time, these values will be used to set the user's process limits.

ulimit (a shell built in in the ksh, bsh, and csh shells) can be used to view and change these values (if you have the authority to change the hard limit).

There are two values that are shown with ulimit, the SOFT limits and the HARD limits.

The hard values are set via the following table:

	3.2		4.1	
Resource	Hard	Soft	Hard	Soft
File Size	fsize	fsize	fsize	fsize
CPU Time	unlimited	cpu	cpu	cpu
Core Size	unlimited	core	unlimited	core
Data Size	unlimited	data	unlimited	data
RSS Size	unlimited	rss	unlimited	rss
Stack Size	unlimited	stack	unlimited	stack

SOFT limits (also called CUR limits):

```
# ulimit -a
time(seconds)    3600
file(blocks)     2097151
data(kbytes)     13073
stack(kbytes)    32768
memory(kbytes)   32768
coredump(blocks) 2048
```

HARD Limits (also called MAX limits);

```
# ulimit -Ha
time(seconds)    unlimited
file(blocks)     2097151
data(kbytes)     245244
stack(kbytes)    248556
memory(kbytes)   unlimited
coredump(blocks) unlimited
```

The SOFT limits can be increased up to the HARD limits, but to increase the HARD limits you need special privileges.

Most of the following information is from the setrlimit(), getrlimit() file, and contributors. The RLIMIT_XXXX definitions come from the setrlimit system call. The units mentioned are the ones that are used to set limits with setrlimit.

Example: If you were using setrlimit() to set the fsize, you would send the value in bytes, but if you were using ksh built-in ulimit you would specify the value in 512-byte blocks.

There is also a ulimit system call that the shell built-ins call. The ulimit referenced in this document is the shell built-in ulimit.

fsize

fsize = 2097151 (units in 512-byte blocks in /etc/security/limits and ulimit)

RLIMIT_FSIZE The largest size, in bytes, of any single file that can be created.

This value is a limit that the kernel will enforce on a user process. This means a user cannot create a file greater than their soft limit unless the root user changes that user's file limit with a "ulimit -Hf" call. The largest number the limit can be set to is 2147483136 bytes, that is, 2Gig. The term "unlimited" could be used in place of 2147483136, but the limit would still be 2Gig.

core

core = 2048 (units in 512-byte blocks in /etc/security/limits and ulimit)

RLIMIT_CORE The largest size, in bytes, of a "core" file that may be created.

This limit is maintained by the kernel and it will not allow a user to create a core file larger than the set SOFT limit. Actually, the system will use the minimum value of the core and fsize SOFT limit.

cpu

cpu = 3600

RLIMIT_CPU The maximum amount of CPU time (in seconds) to be used by each process.

This limit is set (a call is made to setrlimit() by getty) and can be changed, but is not enforced by the kernel. This value may be checked by application code or user code, but the kernel will not enforce this value. In other words, if a process passes its SOFT CPU time limit, the kernel will NOT send a SIGXCPU signal to the offending process (as stated in the documentation, an APAR has been written). The kernel will let processes run as long as they want. So this value can be misleading.

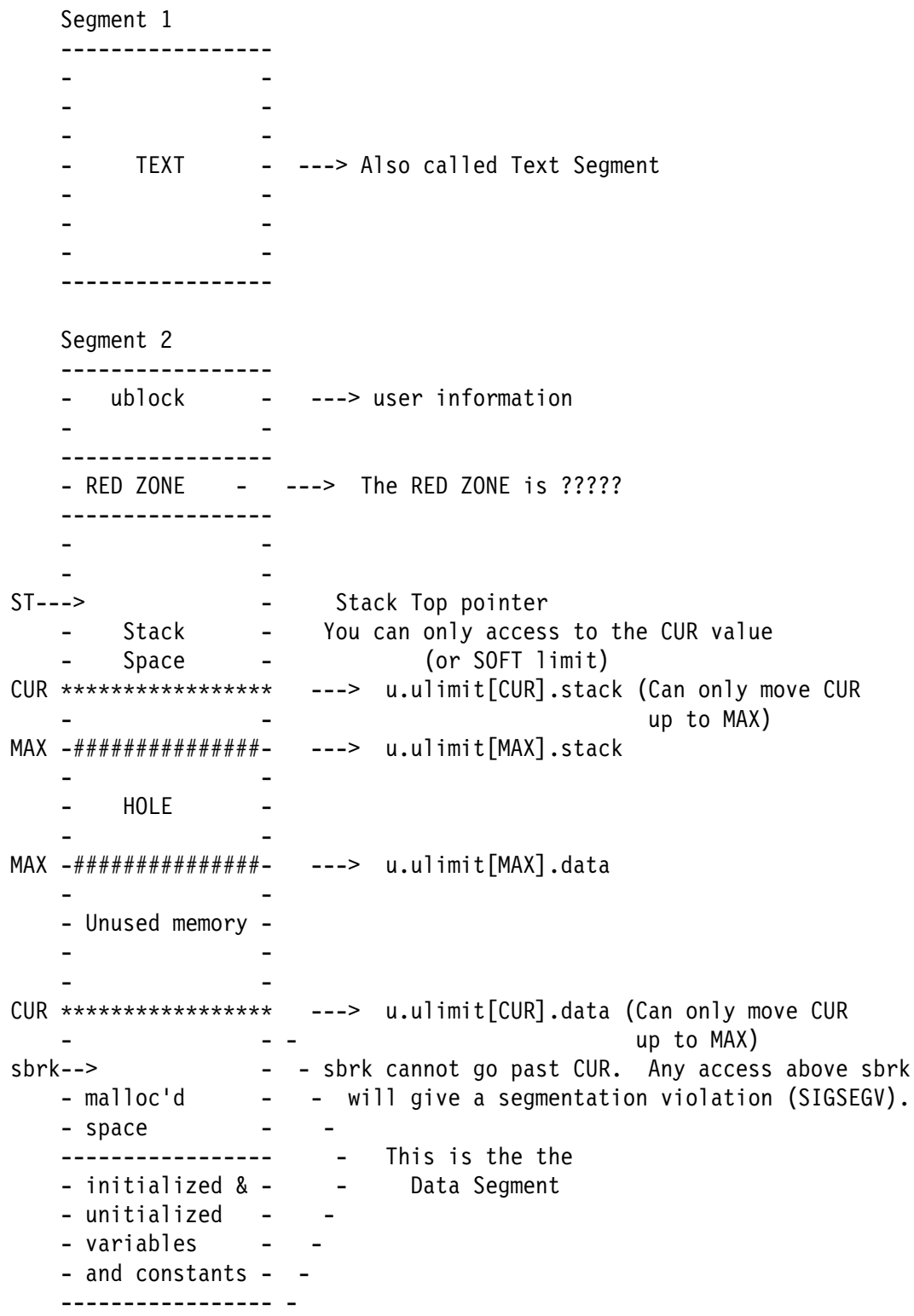
data

data = 131072 (units in 512-byte blocks in /etc/security/limits and kbytes in ulimit)

RLIMIT_DATA The maximum size, in bytes, of the data segment for a process; this defines how far a program may extend its "break" value with the sbrk() system call.

Data and stack are tied together. They exist in Segment 2, and together they can never be greater than approximately 256 MB. This value is actually larger, but 256 MB is a good number to use. You can use the shell ulimit to increase the SOFT/HARD limit for either data or stack, but the HARD limits can never cross. sbrk sets the breakpoint value up to the CUR (SOFT) limit. sbrk moves the breakpoint up and down with an increment value. malloc() calls sbrk.

The following diagram is a representation of how the user space is seen by the kernel.



The user can raise their HARD/SOFT data or stack values with the shell ulimit or setrlimit system call. The kernel will not let the MAXs overlap.

The kernel loader will set the initial sbrk value after it "loads" the initialized and uninitialized variables, but it cannot set the sbrk value past the CUR (SOFT).

The user (or parent process) must set the HARD/SOFT limits higher if more space is needed before the process is executed.

Take the example of a user using the ksh. The user can increase the SOFT limit (up to the HARD limit) of the ksh (with the shell built-in ulimit) before they run a program, because the ksh will be the parent process (it will execute the program) and the program will inherit these limits from the parent process, in this case, the ksh.

With DATA, you might have to lower your STACK value to be able to raise your DATA limit. There is a possiblity that users could kill themselves by lowering their STACK limit lower than what they have already allocated on the stack and then trying to use it, getting a SIGSEGV.

rss

rss = 65536

RLIMIT_RSS The maximum size, in bytes, to which a process's resident set size may grow.

This limit is NOT enforced by the kernel. The kernel will not check this value for a process. It will let the process use as much memory as it needs to run with, and it will not kill a process if it reaches its SOFT limit. The kernel will only start killing processes if system wide paging space gets very low, and that's after it has sent a SIGDANGER signal (init will catch this signal and send a warning message to the console). This happens when the system only has about 2 MB of free paging space, and soon after this it will start killing the youngest process utilizing virtual memory (for AIX 3.2). (NOTE: For AIX 3.1, it will start killing the largest process utilizing virtually memory.)

stack

stack = 8192 (units in 512-byte blocks in /etc/security/limits and kbytes in ulimit)

RLIMIT_STACK The maximum size, in bytes, of the stack segment for a process. This defines how far a program's stack segment may be extended.

A process can access its stack only up to the SOFT limit.

Reader's Comments

Please fax this form to (512) 823-4009, attention "AIXServ Information". You may also e-mail comments to: elizabet@austin.ibm.com. These comments should include the same customer information requested below.

Use this form to tell us what you think about this document. If you have found errors in it, or if you want to express your opinion about it (such as organization, subject matter, appearance) or make suggestions for improvement, this is the form to use.

If you need technical assistance, contact your local branch office, point of sale, or 1-800-CALL-AIX (for information about support offerings). These services may be billable. Faxes on a variety of subjects may be ordered free of charge from 1-800-IBM-4FAX. Outside the U.S. call 415-855-4329 using a fax machine phone.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Note: If you have a problem report or item number, supplying that number may help us determine why a procedure did or did not work in your specific situation.

Problem Report or Item #: Branch Office or Customer #:

Be sure to print your name and fax number below if you would like a reply:

Name: Fax Number:

END OF DOCUMENT (limits.krn, 4FAX# 2461)