How to increase the stack size on MAC OS X

by Eric Circlaeys, Consulting Engineer, Apple Computer Europe (December 17, 2004)

By default Mac OS X' (32-bit Operating System) stack size is **8192KB (8MB)** and the stack storage hard limit available with the default kernel is **65536KB (64MB)**.
You can use **`ulimit`** bash shell built-in command to print/modify the stack size value for the current shell session.

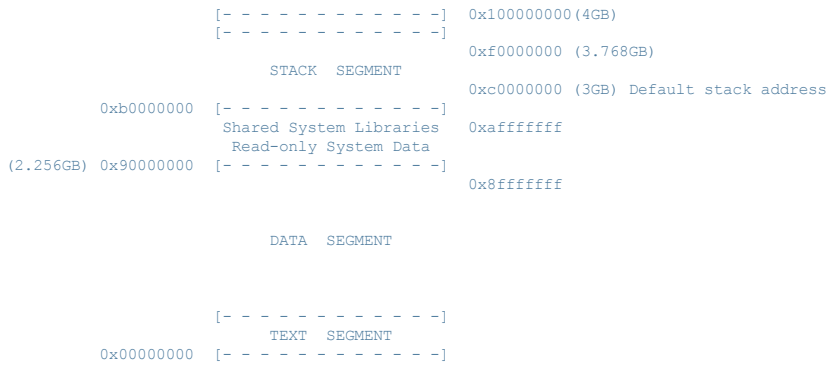
```
$> ulimit -s
8192

$> ulimit -s 65536
or
$> ulimit -s hard

$> ulimit -s
65536
```

Contrary to Mac OS X, Linux systems allow stack size to be as large as the physical memory of the system.
Mac OS X virtual address space is different.

Simplified Mac OS X virtual address space:



In case you need a larger stack for your programming algorithms, there are multiple options to consider.

1. As a first alternative you can recompile the kernel. This case is not a valuable long term solution, because you will have to manage manually system updates to be up-to-date with new Apple releases. Also you break the support options.

You can modify the Darwin (Mac OS X kernel) sources **(`DFLSSIZ` and `MAXSSIZ` defines in `xnu/bsd/ppc/vmparam.h` header file)** to push stack size hard limit up to **256MB**. The **stack grows down** from **0xc0000000** (default stack address) to the first object below in the process virtual address space which ends at **0xb0000000**.
To push the limits up to **1GB (theoretical limit 0xf0000000 - 0xb0000000)**, you will have to modify also the stack address to **0xf0000000 (`USRSTACK` define)**.

Example pushing stack size up to the 256MB limit:

Edit xnu/bsd/ppc/vmparam.h header file from Darwin sources

line: 43
```
#define DFLSSIZ   (128*1024*1024) /* initial stack size limit */
```
line: 46
```
#define MAXSSIZ   (256*1024*1024) /* max stack size */
```

Then recompile the kernel

2. A preferred solution is to **use the GNU linker `ld` options to modify stack size and stack address values** for a specific application without modifying the current default kernel.

GNU Linker `ld` man stack options:

`-stack_addr value`
Specifies the initial address of the stack pointer value, where value is a hexadecimal number rounded to the segment alignment. The default segment alignment is the target pagesize (currently, 1000 hexadecimal for the PowerPC and for i386). If -stack_size is specified and -stack_addr is not, a default stack address specific for the architecture being linked will be used and its value printed as a warning message. This creates a segment named __UNIXSTACK. Note that the initial stack address will be either at the high address of the segment or the low address of the segment depending on which direction the stack grows for the architecture being linked.

`-stack_size value`
Specifies the size of the stack segment value, where value is a hexadecimal number rounded to the segment alignment. The default segment alignment is the target pagesize (currently, 1000 hexadecimal for the PowerPC and for i386). If -stack_addr is specified and -stack_size is not, a default stack size specific for the architecture being linked will be used and its value printed as a warning message. This creates a segment named __UNIXSTACK .

You can directly use these options through **GCC**, **XLC/XLF** IBM compilers or any compilers using GNU linker `ld` with option **`-Wl,-stack_size,value,-stack_addr,value`.**

GCC compiling using stack options (256MB) passed to the linker `ld`:

```
$> gcc source.c -Wl,-stack_size,0x10000000
ld: warning no -stack_addr specified using the default addr: 0xc0000000 (default is 3GB)

$> gcc source.c -Wl,-stack_size,0x10000000,-stack_addr,0xc0000000
```

XLF compiling using stack options (256MB) passed to the linker `ld`:

```
$> xlf source.f -Wl,-stack_size,0x10000000
** _main   === End of Compilation 1 ===
1501-510  Compilation successful for file source.f.
/usr/bin/ld: warning no -stack_addr specified using the default addr: 0xc0000000

$> xlf source.f -Wl,-stack_size,0x10000000,-stack_addr,0xc0000000
** _main   === End of Compilation 1 ===
1501-510  Compilation successful for file source.f.
```

GNU Linker `ld` with stack options (256MB):

```
$> gcc -c source.c
$> ld -stack_size 0x10000000 -lcrt1.o -L/usr/lib/gcc/darwin/3.3 source.o -lgcc -lSystem
ld: warning no -stack_addr specified using the default addr: 0xc0000000
```

GCC compiling using stack options (1GB) passed to the linker `ld`:

```
$> gcc source.c -Wl,-stack_size,0x40000000,-stack_addr,0xf0000000
```

3. If the storage problem is in an XLF-compiled program, an alternative solution is using the **`-qsave`** option to prevent the program from exceeding the stack limit. This will make all local variables STATIC (mapping to heap). Alternatively **`-qnosave`** maps local variables to stack. Remember variables mapped to heap will be retained between calls to a routine (equivalent to the Fortran "`save`"). `XLF` implies `-qsave` by default, `xlf90` not. You have to be aware about reentrance concern.

The following example illustrates the impact of the -qsave option on derived data type:

```
PROGRAM P
  CALL SUB
  CALL SUB
END PROGRAM P

SUBROUTINE SUB
  LOGICAL, SAVE :: FIRST_TIME = .TRUE.
  STRUCTURE /S/
    INTEGER I/17/
  END STRUCTURE
  RECORD /S/ LOCAL_STRUCT
  INTEGER LOCAL_VAR

  IF (FIRST_TIME) THEN
    LOCAL_STRUCT.I = 13
    LOCAL_VAR = 19
    FIRST_TIME = .FALSE.
  ELSE
    ! Prints " 13" if compiled with -qsave or -qsave=all
    ! Prints " 13" if compiled with -qsave=defaultinit
    ! Prints " 17" if compiled with -qnosave
    PRINT *, LOCAL_STRUCT
    ! Prints " 19" if compiled with -qsave or -qsave=all
    ! Value of LOCAL_VAR is undefined otherwise
    PRINT *, LOCAL_VAR
  END IF
END SUBROUTINE SUB
```

4. A cleaner workaround is to eliminate automatic arrays in your code by using allocate statements (or malloc). This does not affect speed.

5. Finally, the future 64 bit version of Mac OS X (Tiger) will be able to manage much larger stack requirements (as there is much less contention for VM space in that case).

Additional Examples:

$> cat source.c

```
/* This code attempts to reach stack limit by large function recursions */

int   recur_call(int *count)
{
  if (*count > 0)
  {
    (*count)--;

    recur_call(count);         /* call recursive function until counter is equal to 0 */
  }
  else
  {
    int i = *count;

    printf("%#x\n", &i);       /* print the last local variable address in the stack reached */
  }
}

int   main(int ac, char **av)
{
  int i = atoi(av[1]);

  printf("%#x\n", &i);         /* print first local variable adress in the stack */

  recur_call(&i);              /* call 'i' times recursive 'recur_call' function, increasing the stack depth */

  printf("Done.\n");

  return (1);
}
```

$> ulimit -s
8192 ***(8MB)***

$> gcc -v source.c -o a.out
```
Reading specs from /usr/libexec/gcc/darwin/ppc/3.3/specs
Thread model: posix
gcc version 3.3 20030304 (Apple Computer, Inc. build 1666)
 /usr/libexec/gcc/darwin/ppc/3.3/cc1 -quiet -v -D__GNUC__=3 -D__GNUC_MINOR__=3 -D__GNUC_PATCHLEVEL__=0 -D__APPLE_CC__=1666
-D__DYNAMIC__ source.c -fPIC -quiet -dumpbase source.c -auxbase a.out -version -o /var/tmp//ccjj4JxH.s
GNU C version 3.3 20030304 (Apple Computer, Inc. build 1666) (ppc-darwin)
    compiled by GNU C version 3.3 20030304 (Apple Computer, Inc. build 1666).
GGC heuristics: --param ggc-min-expand=30 --param ggc-min-heapsize=131072
ignoring nonexistent directory "/usr/ppc-darwin/include"
ignoring nonexistent directory "/Local/Library/Frameworks"
#include "..." search starts here:
#include <...> search starts here:
 /usr/local/include
 /usr/include/gcc/darwin/3.3
 /usr/include
End of search list.
Framework search starts here:
 /System/Library/Frameworks
 /Library/Frameworks
End of framework search list.
 /usr/libexec/gcc/darwin/ppc/as -arch ppc -o /var/tmp//cco2xCem.o /var/tmp//ccjj4JxH.s
 ld -arch ppc -dynamic -o a.out -lcrt1.o -lcrt2.o -L/usr/lib/gcc/darwin/3.3 -L/usr/lib/gcc/darwin -L/usr/libexec/gcc/
darwin/ppc/3.3/../../.. /var/tmp//cco2xCem.o -lgcc -lSystem |
 c++filt
```

$> size a.out
```
__TEXT __DATA __OBJC others dec   hex
4096  4096  0    8192  16384  4000
```

$> ./a.out 10
```
0xbffffc00
0xbffff7e0
Done.
```

$> ./a.out 87354
0xbffffc00 ***(3221224448B)***
0xbf8005e0 ***(3212838368B)***
Done. ***(7MB)***

$> ./a.out 87355
```
0xbffffc00
Segmentation fault
```

$> ulimit -s hard

$> ulimit -s
65536 ***(64MB)***

$> ./a.out 87355
```
0xbffffc00
0xbf800580
Done.
```

$> ./a.out 699023
0xbffffc00 ***(3221224448B)***
0xbc000600 ***(3154118144B)***

```
Done. (63MB)

$> ./a.out 699024
0xbffffc00
Segmentation fault

$> gcc -c source.c

$> ld -o a.out -lcrt1.o -L/usr/lib/gcc/darwin/3.3 source.o -lgcc -lSystem

$> ./a.out 699023
0xbffffc00
0xbc000600
Done.

$> size a.out
__TEXT __DATA __OBJC others dec   hex
4096  4096  0     8192  16384  4000

$> size -m a.out
Segment __PAGEZERO: 4096
Segment __TEXT: 4096
    Section __text: 1952
    Section __picsymbol_stub: 0
    Section __symbol_stub: 0
    Section __picsymbolstub1: 416
    Section __cstring: 164
    total 2532
Segment __DATA: 4096
    Section __data: 32
    Section __la_symbol_ptr: 52
    Section __nl_symbol_ptr: 16
    Section __dyld: 28
    Section __common: 60
    total 188
Segment __LINKEDIT: 4096
total 16384

$> ./a.out 699024
0xbffffc00
Segmentation fault
```

We can increase for example the stack to 70MB = 73400320B = 0x4600000 using `ld` options

```
$> ld -stack_size 0x4600000 -o a.out -lcrt1.o -L/usr/lib/gcc/darwin/3.3 source.o -lgcc -lSystem
ld: warning no -stack_addr specified using the default addr: 0xc0000000 (default is 3GB)

$> ./a.out 699024
0xbffffc00
0xbc0005a0
Done.

$> size a.out
__TEXT __DATA __OBJC others dec   hex
4096  4096  0     73408512    73416704    4604000

$> size -m a.out
Segment __PAGEZERO: 4096
Segment __TEXT: 4096
    Section __text: 2008
    Section __picsymbol_stub: 0
    Section __symbol_stub: 0
    Section __picsymbolstub1: 416
    Section __cstring: 172
    total 2596
Segment __DATA: 4096
    Section __data: 32
    Section __la_symbol_ptr: 52
    Section __nl_symbol_ptr: 16
    Section __dyld: 28
    Section __common: 60
    total 188
Segment __UNIXSTACK: 73400320 (70MB)
Segment __LINKEDIT: 4096
total 73416704
```

We can also use `GCC` options passed to the `ld` linker

```
$> gcc source.c -Wl,-stack_size,0x4600000
ld: warning no -stack_addr specified using the default addr: 0xc0000000 (default is 3GB)

$> ./a.out 699024
0xbffffc00
0xbc0005a0
Done.
```