

# BSDPort jdk7 hotspot changes from linux, os\_cpu

Here are the hotspot os\_cpu changes linux vs bsd:

```
Files src/os_cpu/linux_x86/vm/assembler_linux_x86.cpp and src/os_cpu/bsd_x86/vm/assembler_bsd_x86.cpp are
identical
--- src/os_cpu/linux_x86/vm/atomic_linux_x86.inline.hpp          2011-07-26 20:21:21.000000000 -0600
+++ src/os_cpu/bsd_x86/vm/atomic_bsd_x86.inline.hpp            2011-07-26 20:21:21.000000000 -0600
@@ -22,10 +22,10 @@
 *
 */

-#ifndef OS_CPU_LINUX_X86_VM_ATOMIC_LINUX_X86_INLINE_HPP
-#define OS_CPU_LINUX_X86_VM_ATOMIC_LINUX_X86_INLINE_HPP
+#ifndef OS_CPU_BSD_X86_VM_ATOMIC_BSD_X86_INLINE_HPP
+#define OS_CPU_BSD_X86_VM_ATOMIC_BSD_X86_INLINE_HPP

-#include "orderAccess_linux_x86.inline.hpp"
+#include "orderAccess_bsd_x86.inline.hpp"
#include "runtime/atomic.hpp"
#include "runtime/os.hpp"
#include "vm_version_x86.hpp"
@@ -185,7 +185,7 @@
}

extern "C" {
- // defined in linux_x86.s
+ // defined in bsd_x86.s
    jlong _Atomic_cmpxchg_long(jlong, volatile jlong*, jlong, bool);
    void _Atomic_move_long(volatile jlong* src, volatile jlong* dst);
}
@@ -218,4 +218,4 @@

#endif // AMD64

-#endif // OS_CPU_LINUX_X86_VM_ATOMIC_LINUX_X86_INLINE_HPP
+#endif // OS_CPU_BSD_X86_VM_ATOMIC_BSD_X86_INLINE_HPP
--- src/os_cpu/linux_x86/vm/linux_x86_32.ad          2011-07-26 20:21:21.000000000 -0600
+++ src/os_cpu/bsd_x86/vm/bsd_x86_32.ad            2011-07-26 20:21:21.000000000 -0600
@@ -22,7 +22,7 @@
//
//

-// X86 Linux Architecture Description File
+// X86 Bsd Architecture Description File

//-----OS-DEPENDENT ENCODING BLOCK-----
// This block specifies the encoding classes used by the compiler to output
@@ -53,13 +53,13 @@
// adding a syntax that specifies the sizes of fields in an order,
// so that the adlc can build the emit functions automagically

- enc_class linux_tlsencode (eRegP dst) %{
+ enc_class bsd_tlsencode (eRegP dst) %{
    Register dstReg = as_Register($dst$reg);
    MacroAssembler* masm = new MacroAssembler(&cbuf);
    masm->get_thread(dstReg);
    %}

- enc_class linux_breakpoint %{
+ enc_class bsd_breakpoint %{
    MacroAssembler* masm = new MacroAssembler(&cbuf);
    masm->call(RuntimeAddress(CAST_FROM_FN_PTR(address, os::breakpoint)));
    %}
@@ -108,7 +108,7 @@
effect(DEF dst, KILL cr);

format %{ "MOV    $dst, Thread::current()" %}
```

```

- ins_encode( linux_tlsencode(dst) );
+ ins_encode( bsd_tlsencode(dst) );
ins_pipe( ialu_reg_fat );
%}

@@ -130,7 +130,7 @@
// QQQ TODO for now call breakpoint
// opcode(0xCC);
// ins_encode(Opc);
- ins_encode(linux_breakpoint);
+ ins_encode(bsd_breakpoint);
ins_pipe( pipe_slow );
%}

--- src/os_cpu/linux_x86/vm/linux_x86_32.s          2011-07-26 20:21:21.000000000 -0600
+++ src/os_cpu/bsd_x86/vm/bsd_x86_32.s            2011-07-26 20:21:21.000000000 -0600
@@ -21,6 +21,17 @@
# questions.
#

+
+#ifdef __APPLE__
+# Darwin uses _ prefixed global symbols
+#define SYMBOL(s) _ ## s
+#define ELF_TYPE(name, description)
+#else
+#define SYMBOL(s) s
+#define ELF_TYPE(name, description) .type name,description
+#endif
+
+ .globl SYMBOL(fixcw)

# NOTE WELL! The _Copy functions are called directly
# from server-compiler-generated code via CallLeafNoFP,
@@ -28,42 +39,62 @@
# point or use it in the same manner as does the server
# compiler.

- .globl _Copy_conjoint_bytes
- .globl _Copy_arrayof_conjoint_bytes
- .globl _Copy_conjoint_jshorts_atomic
- .globl _Copy_arrayof_conjoint_jshorts
- .globl _Copy_conjoint_jints_atomic
- .globl _Copy_arrayof_conjoint_jints
- .globl _Copy_conjoint_jlongs_atomic
- .globl _mmx_Copy_arrayof_conjoint_jshorts
+ .globl SYMBOL(_Copy_conjoint_bytes)
+ .globl SYMBOL(_Copy_arrayof_conjoint_bytes)
+ .globl SYMBOL(_Copy_conjoint_jshorts_atomic)
+ .globl SYMBOL(_Copy_arrayof_conjoint_jshorts)
+ .globl SYMBOL(_Copy_conjoint_jints_atomic)
+ .globl SYMBOL(_Copy_arrayof_conjoint_jints)
+ .globl SYMBOL(_Copy_conjoint_jlongs_atomic)
+ .globl SYMBOL(_mmx_Copy_arrayof_conjoint_jshorts)

- .globl _Atomic_cmpxchg_long
- .globl _Atomic_move_long
+ .globl SYMBOL(_Atomic_cmpxchg_long)
+ .globl SYMBOL(_Atomic_move_long)

.text

- .globl SafeFetch32, Fetch32PFI, Fetch32Resume
- .globl SafeFetchN
+# Support for void os::Solaris::init_thread_fpu_state() in os_solaris_i486.cpp
+# Set fpu to 53 bit precision. This happens too early to use a stub.
+# ported from solaris_x86_32.s
+#ifdef __APPLE__
+ .align 4
+#else
+ .align 16

```

```

+endif
+SYMBOL(fixcw):
+    pushl    $0x27f
+    fldcw   0(%esp)
+    popl    %eax
+    ret
+
+#ifdef __APPLE__
+    .align  4
+#else
+    .align  16
+#endif
+
+.globl  SYMBOL(SafeFetch32), SYMBOL(Fetch32PFI), SYMBOL(Fetch32Resume)
+.globl  SYMBOL(SafeFetchN)
## TODO: avoid exposing Fetch32PFI and Fetch32Resume.
## Instead, the signal handler would call a new SafeFetchTriage(FaultingEIP)
## routine to vet the address.  If the address is the faulting LD then
## SafeFetchTriage() would return the resume-at EIP, otherwise null.
-    .type   SafeFetch32,@function
+    ELF_TYPE(SafeFetch32,@function)
    .p2align 4,,15
-SafeFetch32:
-SafeFetchN:
+SYMBOL(SafeFetch32):
+SYMBOL(SafeFetchN):
    movl    0x8(%esp), %eax
    movl    0x4(%esp), %ecx
-Fetch32PFI:
+SYMBOL(Fetch32PFI):
    movl    (%ecx), %eax
-Fetch32Resume:
+SYMBOL(Fetch32Resume):
    ret

-    .globl  SpinPause
-    .type   SpinPause,@function
+    .globl  SYMBOL(SpinPause)
+    ELF_TYPE(SpinPause,@function)
    .p2align 4,,15
-SpinPause:
+SYMBOL(SpinPause):
    rep
    nop
    movl    $1, %eax
@@ -73,8 +104,8 @@
    #
    #                                     void* to,
    #                                     size_t count)
    .p2align 4,,15
-    .type   _Copy_conjoint_bytes,@function
-_Copy_conjoint_bytes:
+    ELF_TYPE(_Copy_conjoint_bytes,@function)
+SYMBOL(_Copy_conjoint_bytes):
    pushl   %esi
    movl    4+12(%esp),%ecx    # count
    pushl   %edi
@@ -181,8 +212,8 @@
    #
    # Same as _Copy_conjoint_bytes, except no source alignment check.
    .p2align 4,,15
-    .type   _Copy_arrayof_conjoint_bytes,@function
-_Copy_arrayof_conjoint_bytes:
+    ELF_TYPE(_Copy_arrayof_conjoint_bytes,@function)
+SYMBOL(_Copy_arrayof_conjoint_bytes):
    pushl   %esi
    movl    4+12(%esp),%ecx    # count
    pushl   %edi
@@ -269,8 +300,8 @@
    #
    #                                     void* to,
    #                                     size_t count)

```

```

        .p2align 4,,15
-       .type      _Copy_conjoint_jshorts_atomic,@function
- _Copy_conjoint_jshorts_atomic:
+       ELF_TYPE(_Copy_conjoint_jshorts_atomic,@function)
+SYMBOL(_Copy_conjoint_jshorts_atomic):
        pushl    %esi
        movl    4+12(%esp),%ecx    # count
        pushl    %edi
@@ -356,8 +387,8 @@
        #
        #                               void* to,
        #                               size_t count)
        .p2align 4,,15
-       .type      _Copy_arrayof_conjoint_jshorts,@function
- _Copy_arrayof_conjoint_jshorts:
+       ELF_TYPE(_Copy_arrayof_conjoint_jshorts,@function)
+SYMBOL(_Copy_arrayof_conjoint_jshorts):
        pushl    %esi
        movl    4+12(%esp),%ecx    # count
        pushl    %edi
@@ -433,10 +464,10 @@
        # Equivalent to
        #   arrayof_conjoint_jints
        .p2align 4,,15
-       .type      _Copy_conjoint_jints_atomic,@function
-       .type      _Copy_arrayof_conjoint_jints,@function
- _Copy_conjoint_jints_atomic:
- _Copy_arrayof_conjoint_jints:
+       ELF_TYPE(_Copy_conjoint_jints_atomic,@function)
+       ELF_TYPE(_Copy_arrayof_conjoint_jints,@function)
+SYMBOL(_Copy_conjoint_jints_atomic):
+SYMBOL(_Copy_arrayof_conjoint_jints):
        pushl    %esi
        movl    4+12(%esp),%ecx    # count
        pushl    %edi
@@ -498,7 +529,7 @@
        #
        # count treated as signed
        #
-       # if (from > to) {
+       # // if (from > to) {
        #   while (--count >= 0) {
        #     *to++ = *from++;
        #   }
@@ -508,8 +539,8 @@
        # }
        # }
        .p2align 4,,15
-       .type      _Copy_conjoint_jlongs_atomic,@function
- _Copy_conjoint_jlongs_atomic:
+       ELF_TYPE(_Copy_conjoint_jlongs_atomic,@function)
+SYMBOL(_Copy_conjoint_jlongs_atomic):
        movl    4+8(%esp),%ecx    # count
        movl    4+0(%esp),%eax    # from
        movl    4+4(%esp),%edx    # to
@@ -537,8 +568,8 @@
        #
        #                               void* to,
        #                               size_t count)
        .p2align 4,,15
-       .type      _mmx_Copy_arrayof_conjoint_jshorts,@function
- _mmx_Copy_arrayof_conjoint_jshorts:
+       ELF_TYPE(_mmx_Copy_arrayof_conjoint_jshorts,@function)
+SYMBOL(_mmx_Copy_arrayof_conjoint_jshorts):
        pushl    %esi
        movl    4+12(%esp),%ecx
        pushl    %edi
@@ -636,8 +667,8 @@
        #
        #                               bool is_MP)
        #
        .p2align 4,,15
-       .type      _Atomic_cmpxchg_long,@function
- _Atomic_cmpxchg_long:

```

```

+         ELF_TYPE(_Atomic_cmpxchg_long,@function)
+SYMBOL(_Atomic_cmpxchg_long):
           pushl    %ebx           # 8(%esp) : return PC
           pushl    %edi           # 4(%esp) : old %ebx
           # 0(%esp) : old %edi
--- src/os_cpu/linux_x86/vm/linux_x86_64.ad      2011-07-26 20:21:21.000000000 -0600
+++ src/os_cpu/bsd_x86/vm/bsd_x86_64.ad        2011-07-26 20:21:21.000000000 -0600
@@ -22,7 +22,7 @@
//
//

-// AMD64 Linux Architecture Description File
+// AMD64 BSD Architecture Description File

//-----OS-DEPENDENT ENCODING BLOCK-----
// This block specifies the encoding classes used by the compiler to
@@ -70,7 +70,7 @@
emit_opcode(cbuf, 0xD0 | (R10_enc - 8));
%}

- enc_class linux_breakpoint
+ enc_class bsd_breakpoint
%{
    MacroAssembler* masm = new MacroAssembler(&cbuf);
    masm->call(RuntimeAddress(CAST_FROM_FN_PTR(address, os::breakpoint)));
@@ -141,7 +141,7 @@
// QQQ TODO for now call breakpoint
// opcode(0xCC);
// ins_encode(Opc);
- ins_encode(linux_breakpoint);
+ ins_encode(bsd_breakpoint);
ins_pipe(pipe_slow);
%}

--- src/os_cpu/linux_x86/vm/linux_x86_64.s      2011-07-26 20:21:21.000000000 -0600
+++ src/os_cpu/bsd_x86/vm/bsd_x86_64.s        2011-07-26 20:21:21.000000000 -0600
@@ -21,6 +21,14 @@
# questions.
#

+#ifdef __APPLE__
+# Darwin uses _ prefixed global symbols
+#define SYMBOL(s) _ ## s
+#define ELF_TYPE(name, description)
+#else
+#define SYMBOL(s) s
+#define ELF_TYPE(name, description) .type name,description
+#endif

# NOTE WELL! The _Copy functions are called directly
# from server-compiler-generated code via CallLeafNoFP,
@@ -28,42 +36,54 @@
# point or use it in the same manner as does the server
# compiler.

- .globl _Copy_arrayof_conjoint_bytes
- .globl _Copy_arrayof_conjoint_jshorts
- .globl _Copy_conjoint_jshorts_atomic
- .globl _Copy_arrayof_conjoint_jints
- .globl _Copy_conjoint_jints_atomic
- .globl _Copy_arrayof_conjoint_jlongs
- .globl _Copy_conjoint_jlongs_atomic
+ .globl SYMBOL(_Copy_arrayof_conjoint_bytes)
+ .globl SYMBOL(_Copy_arrayof_conjoint_jshorts)
+ .globl SYMBOL(_Copy_conjoint_jshorts_atomic)
+ .globl SYMBOL(_Copy_arrayof_conjoint_jints)
+ .globl SYMBOL(_Copy_conjoint_jints_atomic)
+ .globl SYMBOL(_Copy_arrayof_conjoint_jlongs)
+ .globl SYMBOL(_Copy_conjoint_jlongs_atomic)

.text

```

```

-     .globl SafeFetch32, Fetch32PFI, Fetch32Resume
+     .globl SYMBOL(SafeFetch32), SYMBOL(Fetch32PFI), SYMBOL(Fetch32Resume)
+#ifdef __APPLE__
+     .align 4
+#else
+     .align 16
-     .type SafeFetch32,@function
+#endif
+     ELF_TYPE(SafeFetch32,@function)
+     // Prototype: int SafeFetch32 (int * Adr, int ErrValue)
-SafeFetch32:
+SYMBOL(SafeFetch32):
+     movl    %esi, %eax
-Fetch32PFI:
+SYMBOL(Fetch32PFI):
+     movl    (%rdi), %eax
-Fetch32Resume:
+SYMBOL(Fetch32Resume):
+     ret

-     .globl SafeFetchN, FetchNPFI, FetchNResume
+     .globl SYMBOL(SafeFetchN), SYMBOL(FetchNPFI), SYMBOL(FetchNResume)
+#ifdef __APPLE__
+     .align 4
+#else
+     .align 16
-     .type SafeFetchN,@function
+#endif
+     ELF_TYPE(SafeFetchN,@function)
+     // Prototype: intptr_t SafeFetchN (intptr_t * Adr, intptr_t ErrValue)
-SafeFetchN:
+SYMBOL(SafeFetchN):
+     movq    %rsi, %rax
-FetchNPFI:
+SYMBOL(FetchNPFI):
+     movq    (%rdi), %rax
-FetchNResume:
+SYMBOL(FetchNResume):
+     ret

-     .globl SpinPause
-     .align 16
-     .type SpinPause,@function
-SpinPause:
+     .globl SYMBOL(SpinPause)
+#ifdef __APPLE__
+     .align 4
+#else
+     .align 16
+#endif
+     ELF_TYPE(SpinPause,@function)
+SYMBOL(SpinPause):
+     rep
+     nop
+     movq    $1, %rax
@@ -77,8 +97,8 @@
+     # rdx - count, treated as ssize_t
+     #
+     .p2align 4,,15
-     .type _Copy_arrayof_conjoint_bytes,@function
-_Copy_arrayof_conjoint_bytes:
+     ELF_TYPE(_Copy_arrayof_conjoint_bytes,@function)
+SYMBOL(_Copy_arrayof_conjoint_bytes):
+     movq    %rdx,%r8           # byte count
+     shrq    $3,%rdx          # qword count
+     cmpq    %rdi,%rsi
@@ -179,10 +199,10 @@
+     # rdx - count, treated as ssize_t
+     #
+     .p2align 4,,15

```

```

-         .type    _Copy_arrayof_conjoint_jshorts,@function
-         .type    _Copy_conjoint_jshorts_atomic,@function
- _Copy_arrayof_conjoint_jshorts:
- _Copy_conjoint_jshorts_atomic:
+         ELF_TYPE(_Copy_arrayof_conjoint_jshorts,@function)
+         ELF_TYPE(_Copy_conjoint_jshorts_atomic,@function)
+SYMBOL(_Copy_arrayof_conjoint_jshorts):
+SYMBOL(_Copy_conjoint_jshorts_atomic):
         movq    %rdx,%r8          # word count
         shrq    $2,%rdx          # qword count
         cmpq    %rdi,%rsi
@@ -269,10 +289,10 @@
         # rdx - count, treated as ssize_t
         #
         .p2align 4,,15
-         .type    _Copy_arrayof_conjoint_jints,@function
-         .type    _Copy_conjoint_jints_atomic,@function
- _Copy_arrayof_conjoint_jints:
- _Copy_conjoint_jints_atomic:
+         ELF_TYPE(_Copy_arrayof_conjoint_jints,@function)
+         ELF_TYPE(_Copy_conjoint_jints_atomic,@function)
+SYMBOL(_Copy_arrayof_conjoint_jints):
+SYMBOL(_Copy_conjoint_jints_atomic):
         movq    %rdx,%r8          # dword count
         shrq    %rdx            # qword count
         cmpq    %rdi,%rsi
@@ -348,10 +368,10 @@
         # rdx - count, treated as ssize_t
         #
         .p2align 4,,15
-         .type    _Copy_arrayof_conjoint_jlongs,@function
-         .type    _Copy_conjoint_jlongs_atomic,@function
- _Copy_arrayof_conjoint_jlongs:
- _Copy_conjoint_jlongs_atomic:
+         ELF_TYPE(_Copy_arrayof_conjoint_jlongs,@function)
+         ELF_TYPE(_Copy_conjoint_jlongs_atomic,@function)
+SYMBOL(_Copy_arrayof_conjoint_jlongs):
+SYMBOL(_Copy_conjoint_jlongs_atomic):
         cmpq    %rdi,%rsi
         leaq    -8(%rdi,%rdx,8),%rax # from + count*8 - 8
         jbe    acl_CopyRight
--- src/os_cpu/linux_x86/vm/bytes_linux_x86.inline.hpp          2011-07-26 20:21:21.000000000 -0600
+++ src/os_cpu/bsd_x86/vm/bytes_bsd_x86.inline.hpp             2011-07-26 20:21:21.000000000 -0600
@@ -22,10 +22,36 @@
*
*/

-#ifndef OS_CPU_LINUX_X86_VM_BYTES_LINUX_X86_INLINE_HPP
-#define OS_CPU_LINUX_X86_VM_BYTES_LINUX_X86_INLINE_HPP
+#ifndef OS_CPU_BSD_X86_VM_BYTES_BSD_X86_INLINE_HPP
+#define OS_CPU_BSD_X86_VM_BYTES_BSD_X86_INLINE_HPP

+#ifndef _ALLBSD_SOURCE
+ #include <byteswap.h>
+#endif
+
+#ifdef __APPLE__
+#include <libkern/OSByteOrder.h>
+#endif
+
+#if defined(AMD64)
+# if defined(__APPLE__)
+#  define bswap_16(x)          OSSwapInt16(x)
+#  define bswap_32(x)          OSSwapInt32(x)
+#  define bswap_64(x)          OSSwapInt64(x)
+# elif defined(__OpenBSD__)
+#  define bswap_16(x)          swap16(x)
+#  define bswap_32(x)          swap32(x)
+#  define bswap_64(x)          swap64(x)
+# elif defined(__NetBSD__)
+#  define bswap_16(x)          bswap16(x)

```

```

+#   define bswap_32(x)          bswap32(x)
+#   define bswap_64(x)          bswap64(x)
+#   else
+#   define bswap_16(x)  __bswap16(x)
+#   define bswap_32(x)  __bswap32(x)
+#   define bswap_64(x)  __bswap64(x)
+#   endif
+#endif

// Efficient swapping of data bytes from Java byte
// ordering to native byte ordering and vice versa.
@@ -87,4 +113,4 @@
}
#endif // !AMD64

-#endif // OS_CPU_LINUX_X86_VM_BYTES_LINUX_X86_INLINE_HPP
+#endif // OS_CPU_BSD_X86_VM_BYTES_BSD_X86_INLINE_HPP
--- src/os_cpu/linux_x86/vm/copy_linux_x86.inline.hpp          2011-07-26 20:21:21.000000000 -0600
+++ src/os_cpu/bsd_x86/vm/copy_bsd_x86.inline.hpp            2011-07-26 20:21:21.000000000 -0600
@@ -22,8 +22,8 @@
*
*/

-#ifndef OS_CPU_LINUX_X86_VM_COPY_LINUX_X86_INLINE_HPP
-#define OS_CPU_LINUX_X86_VM_COPY_LINUX_X86_INLINE_HPP
+#ifndef OS_CPU_BSD_X86_VM_COPY_BSD_X86_INLINE_HPP
+#define OS_CPU_BSD_X86_VM_COPY_BSD_X86_INLINE_HPP

static void pd_conjoint_words(HeapWord* from, HeapWord* to, size_t count) {
#ifdef AMD64
@@ -306,4 +306,4 @@
#endif // AMD64
}

-#endif // OS_CPU_LINUX_X86_VM_COPY_LINUX_X86_INLINE_HPP
+#endif // OS_CPU_BSD_X86_VM_COPY_BSD_X86_INLINE_HPP
--- src/os_cpu/linux_x86/vm/globals_linux_x86.hpp            2011-07-26 20:21:21.000000000 -0600
+++ src/os_cpu/bsd_x86/vm/globals_bsd_x86.hpp                2011-07-26 20:21:21.000000000 -0600
@@ -22,12 +22,13 @@
*
*/

-#ifndef OS_CPU_LINUX_X86_VM_GLOBALS_LINUX_X86_HPP
-#define OS_CPU_LINUX_X86_VM_GLOBALS_LINUX_X86_HPP
+#ifndef OS_CPU_BSD_X86_VM_GLOBALS_BSD_X86_HPP
+#define OS_CPU_BSD_X86_VM_GLOBALS_BSD_X86_HPP

+//
// Sets the default values for platform dependent flags used by the runtime system.
// (see globals.hpp)
-
+//
define_pd_global(bool, DontYieldALot,          false);
#ifdef AMD64
define_pd_global(intx, ThreadStackSize,        1024); // 0 => use system default
@@ -41,12 +42,13 @@
#endif // AMD64

define_pd_global(intx, CompilerThreadStackSize, 0);
+define_pd_global(intx, SurvivorRatio,         8);

-define_pd_global(uintx, JVMInvokeMethodSlack, 8192);
+define_pd_global(uintx, JVMInvokeMethodSlack, 8192);

// Only used on 64 bit platforms
-define_pd_global(uintx, HeapBaseMinAddress,    2*G);
+define_pd_global(uintx, HeapBaseMinAddress,    2*G);
// Only used on 64 bit Windows platforms
define_pd_global(bool, UseVectoredExceptions,   false);

-#endif // OS_CPU_LINUX_X86_VM_GLOBALS_LINUX_X86_HPP

```



```

+#endif // OS_CPU_BSD_X86_VM_GLOBALS_BSD_X86_HPP
--- src/os_cpu/linux_x86/vm/orderAccess_linux_x86.inline.hpp          2011-07-26 20:21:21.000000000 -0600
+++ src/os_cpu/bsd_x86/vm/orderAccess_bsd_x86.inline.hpp           2011-07-26 20:21:21.000000000 -0600
@@ -22,8 +22,8 @@
*
*/

-#ifndef OS_CPU_LINUX_X86_VM_ORDERACCESS_LINUX_X86_INLINE_HPP
-#define OS_CPU_LINUX_X86_VM_ORDERACCESS_LINUX_X86_INLINE_HPP
+#ifndef OS_CPU_BSD_X86_VM_ORDERACCESS_BSD_X86_INLINE_HPP
+#define OS_CPU_BSD_X86_VM_ORDERACCESS_BSD_X86_INLINE_HPP

#include "runtime/atomic.hpp"
#include "runtime/orderAccess.hpp"
@@ -212,4 +212,4 @@
#endif // AMD64
}

-#endif // OS_CPU_LINUX_X86_VM_ORDERACCESS_LINUX_X86_INLINE_HPP
+#endif // OS_CPU_BSD_X86_VM_ORDERACCESS_BSD_X86_INLINE_HPP
--- src/os_cpu/linux_x86/vm/os_linux_x86.cpp          2011-07-26 20:21:21.000000000 -0600
+++ src/os_cpu/bsd_x86/vm/os_bsd_x86.cpp            2011-07-26 20:21:21.000000000 -0600
@@ -30,11 +30,11 @@
#include "code/icBuffer.hpp"
#include "code/vtableStubs.hpp"
#include "interpreter/interpreter.hpp"
-#include "jvm_linux.h"
+#include "jvm_bsd.h"
#include "memory/allocation.inline.hpp"
-#include "mutex_linux.inline.hpp"
+#include "mutex_bsd.inline.hpp"
#include "nativeInst_x86.hpp"
-#include "os_share_linux.hpp"
+#include "os_share_bsd.hpp"
#include "prims/jniFastGetField.hpp"
#include "prims/jvm.h"
#include "prims/jvm_misc.hpp"
@@ -49,7 +49,7 @@
#include "runtime/sharedRuntime.hpp"
#include "runtime/stubRoutines.hpp"
#include "runtime/timer.hpp"
-#include "thread_linux.inline.hpp"
+#include "thread_bsd.inline.hpp"
#include "utilities/events.hpp"
#include "utilities/vmError.hpp"
#ifdef COMPILER1
@@ -78,23 +78,202 @@
#include <sys/wait.h>
#include <pwd.h>
#include <poll.h>
+#ifndef __OpenBSD__
#include <ucontext.h>
-#include <fpu_control.h>
+#endif
+
+
+#if defined(_ALLBSD_SOURCE) && !defined(__APPLE__) && !defined(__NetBSD__)
+#include <pthread_np.h>
+#endif

#ifdef AMD64
-#define REG_SP REG_RSP
-#define REG_PC REG_RIP
-#define REG_FP REG_RBP
#define SPELL_REG_SP "rsp"
#define SPELL_REG_FP "rbp"
#else
-#define REG_SP REG_UESP
-#define REG_PC REG_EIP
-#define REG_FP REG_EBP
#define SPELL_REG_SP "esp"
#define SPELL_REG_FP "ebp"

```

```

#endif // AMD64

+#ifdef __FreeBSD__
+# define context_trapno uc_mcontext.mc_trapno
+# ifdef AMD64
+# define context_pc uc_mcontext.mc_rip
+# define context_sp uc_mcontext.mc_rsp
+# define context_fp uc_mcontext.mc_rbp
+# define context_rip uc_mcontext.mc_rip
+# define context_rsp uc_mcontext.mc_rsp
+# define context_rbp uc_mcontext.mc_rbp
+# define context_rax uc_mcontext.mc_rax
+# define context_rbx uc_mcontext.mc_rbx
+# define context_rcx uc_mcontext.mc_rcx
+# define context_rdx uc_mcontext.mc_rdx
+# define context_rsi uc_mcontext.mc_rsi
+# define context_rdi uc_mcontext.mc_rdi
+# define context_r8 uc_mcontext.mc_r8
+# define context_r9 uc_mcontext.mc_r9
+# define context_r10 uc_mcontext.mc_r10
+# define context_r11 uc_mcontext.mc_r11
+# define context_r12 uc_mcontext.mc_r12
+# define context_r13 uc_mcontext.mc_r13
+# define context_r14 uc_mcontext.mc_r14
+# define context_r15 uc_mcontext.mc_r15
+# define context_flags uc_mcontext.mc_flags
+# define context_err uc_mcontext.mc_err
+# else
+# define context_pc uc_mcontext.mc_eip
+# define context_sp uc_mcontext.mc_esp
+# define context_fp uc_mcontext.mc_ebp
+# define context_eip uc_mcontext.mc_eip
+# define context_esp uc_mcontext.mc_esp
+# define context_eax uc_mcontext.mc_eax
+# define context_ebx uc_mcontext.mc_ebx
+# define context_ecx uc_mcontext.mc_ecx
+# define context_edx uc_mcontext.mc_edx
+# define context_ebp uc_mcontext.mc_ebp
+# define context_esi uc_mcontext.mc_esi
+# define context_edi uc_mcontext.mc_edi
+# define context_eflags uc_mcontext.mc_eflags
+# define context_trapno uc_mcontext.mc_trapno
+# endif
+#endif
+
+#ifdef __APPLE__
+# if __DARWIN_UNIX03 && (MAC_OS_X_VERSION_MAX_ALLOWED >= MAC_OS_X_VERSION_10_5)
+ // 10.5 UNIX03 member name prefixes
+ #define DU3_PREFIX(s, m) __ ## s. ## m
+# else
+ #define DU3_PREFIX(s, m) s ## . ## m
+# endif
+
+# ifdef AMD64
+# define context_pc context_rip
+# define context_sp context_rsp
+# define context_fp context_rbp
+# define context_rip uc_mcontext->DU3_PREFIX(ss,rip)
+# define context_rsp uc_mcontext->DU3_PREFIX(ss,rsp)
+# define context_rax uc_mcontext->DU3_PREFIX(ss,rax)
+# define context_rbx uc_mcontext->DU3_PREFIX(ss,rbx)
+# define context_rcx uc_mcontext->DU3_PREFIX(ss,rcx)
+# define context_rdx uc_mcontext->DU3_PREFIX(ss,rdx)
+# define context_rbp uc_mcontext->DU3_PREFIX(ss,rbp)
+# define context_rsi uc_mcontext->DU3_PREFIX(ss,rsi)
+# define context_rdi uc_mcontext->DU3_PREFIX(ss,rdi)
+# define context_r8 uc_mcontext->DU3_PREFIX(ss,r8)
+# define context_r9 uc_mcontext->DU3_PREFIX(ss,r9)
+# define context_r10 uc_mcontext->DU3_PREFIX(ss,r10)
+# define context_r11 uc_mcontext->DU3_PREFIX(ss,r11)
+# define context_r12 uc_mcontext->DU3_PREFIX(ss,r12)

```

```

+# define context_r13 uc_mcontext->DU3_PREFIX(ss,r13)
+# define context_r14 uc_mcontext->DU3_PREFIX(ss,r14)
+# define context_r15 uc_mcontext->DU3_PREFIX(ss,r15)
+# define context_flags uc_mcontext->DU3_PREFIX(ss,rflags)
+# define context_trapno uc_mcontext->DU3_PREFIX(es,trapno)
+# define context_err uc_mcontext->DU3_PREFIX(es,err)
+# else
+# define context_pc context_eip
+# define context_sp context_esp
+# define context_fp context_ebp
+# define context_eip uc_mcontext->DU3_PREFIX(ss,eip)
+# define context_esp uc_mcontext->DU3_PREFIX(ss,esp)
+# define context_eax uc_mcontext->DU3_PREFIX(ss,eax)
+# define context_ebx uc_mcontext->DU3_PREFIX(ss,ebx)
+# define context_ecx uc_mcontext->DU3_PREFIX(ss,ecx)
+# define context_edx uc_mcontext->DU3_PREFIX(ss,edx)
+# define context_ebp uc_mcontext->DU3_PREFIX(ss,ebp)
+# define context_esi uc_mcontext->DU3_PREFIX(ss,esi)
+# define context_edi uc_mcontext->DU3_PREFIX(ss,edi)
+# define context_eflags uc_mcontext->DU3_PREFIX(ss,eflags)
+# define context_trapno uc_mcontext->DU3_PREFIX(es,trapno)
+# endif
+#endif
+
+#ifdef __OpenBSD__
+# define context_trapno sc_trapno
+# ifdef AMD64
+# define context_pc sc_rip
+# define context_sp sc_rsp
+# define context_fp sc_rbp
+# define context_rip sc_rip
+# define context_rsp sc_rsp
+# define context_rbp sc_rbp
+# define context_rax sc_rax
+# define context_rbx sc_rbx
+# define context_rcx sc_rcx
+# define context_rdx sc_rdx
+# define context_rsi sc_rsi
+# define context_rdi sc_rdi
+# define context_r8 sc_r8
+# define context_r9 sc_r9
+# define context_r10 sc_r10
+# define context_r11 sc_r11
+# define context_r12 sc_r12
+# define context_r13 sc_r13
+# define context_r14 sc_r14
+# define context_r15 sc_r15
+# define context_flags sc_rflags
+# define context_err sc_err
+# else
+# define context_pc sc_eip
+# define context_sp sc_esp
+# define context_fp sc_ebp
+# define context_eip sc_eip
+# define context_esp sc_esp
+# define context_eax sc_eax
+# define context_ebx sc_ebx
+# define context_ecx sc_ecx
+# define context_edx sc_edx
+# define context_ebp sc_ebp
+# define context_esi sc_esi
+# define context_edi sc_edi
+# define context_eflags sc_eflags
+# define context_trapno sc_trapno
+# endif
+#endif
+
+#ifdef __NetBSD__
+# define context_trapno uc_mcontext.__gregs[_REG_TRAPNO]
+# ifdef AMD64
+# define __register_t __greg_t

```

```

+# define context_pc uc_mcontext.__gregs[_REG_RIP]
+# define context_sp uc_mcontext.__gregs[_REG_URSP]
+# define context_fp uc_mcontext.__gregs[_REG_RBP]
+# define context_rip uc_mcontext.__gregs[_REG_RIP]
+# define context_rsp uc_mcontext.__gregs[_REG_URSP]
+# define context_rax uc_mcontext.__gregs[_REG_RAX]
+# define context_rbx uc_mcontext.__gregs[_REG_RBX]
+# define context_rcx uc_mcontext.__gregs[_REG_RCX]
+# define context_rdx uc_mcontext.__gregs[_REG_RDX]
+# define context_rbp uc_mcontext.__gregs[_REG_RBP]
+# define context_rsi uc_mcontext.__gregs[_REG_RSI]
+# define context_rdi uc_mcontext.__gregs[_REG_RDI]
+# define context_r8 uc_mcontext.__gregs[_REG_R8]
+# define context_r9 uc_mcontext.__gregs[_REG_R9]
+# define context_r10 uc_mcontext.__gregs[_REG_R10]
+# define context_r11 uc_mcontext.__gregs[_REG_R11]
+# define context_r12 uc_mcontext.__gregs[_REG_R12]
+# define context_r13 uc_mcontext.__gregs[_REG_R13]
+# define context_r14 uc_mcontext.__gregs[_REG_R14]
+# define context_r15 uc_mcontext.__gregs[_REG_R15]
+# define context_flags uc_mcontext.__gregs[_REG_RFL]
+# define context_err uc_mcontext.__gregs[_REG_ERR]
+# else
+# define context_pc uc_mcontext.__gregs[_REG_EIP]
+# define context_sp uc_mcontext.__gregs[_REG_UESP]
+# define context_fp uc_mcontext.__gregs[_REG_EBP]
+# define context_eip uc_mcontext.__gregs[_REG_EIP]
+# define context_esp uc_mcontext.__gregs[_REG_UESP]
+# define context_eax uc_mcontext.__gregs[_REG_EAX]
+# define context_ebx uc_mcontext.__gregs[_REG_EBX]
+# define context_ecx uc_mcontext.__gregs[_REG_ECX]
+# define context_edx uc_mcontext.__gregs[_REG_EDX]
+# define context_ebp uc_mcontext.__gregs[_REG_EBP]
+# define context_esi uc_mcontext.__gregs[_REG_ESI]
+# define context_edi uc_mcontext.__gregs[_REG_EDI]
+# define context_eflags uc_mcontext.__gregs[_REG_EFL]
+# define context_trapno uc_mcontext.__gregs[_REG_TRAPNO]
+# endif
+#endif
+
+ address os::current_stack_pointer() {
+ #ifdef SPARC_WORKS
+ register void *esp;
+ @@ -118,24 +297,24 @@
+ // Nothing to do.
+ }

-address os::Linux::ucontext_get_pc(ucontext_t * uc) {
- return (address)uc->uc_mcontext.gregs[REG_PC];
+address os::Bsd::ucontext_get_pc(ucontext_t * uc) {
+ return (address)uc->context_pc;
+ }

-intptr_t* os::Linux::ucontext_get_sp(ucontext_t * uc) {
- return (intptr_t*)uc->uc_mcontext.gregs[REG_SP];
+intptr_t* os::Bsd::ucontext_get_sp(ucontext_t * uc) {
+ return (intptr_t*)uc->context_sp;
+ }

-intptr_t* os::Linux::ucontext_get_fp(ucontext_t * uc) {
- return (intptr_t*)uc->uc_mcontext.gregs[REG_FP];
+intptr_t* os::Bsd::ucontext_get_fp(ucontext_t * uc) {
+ return (intptr_t*)uc->context_fp;
+ }

+ // For Forte Analyzer AsyncGetCallTrace profiling support - thread
+ // is currently interrupted by SIGPROF.
+ // os::Solaris::fetch_frame_from_ucontext() tries to skip nested signal
+ // frames. Currently we don't do that on Linux, so it's the same as
+ // frames. Currently we don't do that on Bsd, so it's the same as
+ // os::fetch_frame_from_context().

```

```

-ExtendedPC os::Linux::fetch_frame_from_ucontext(Thread* thread,
+ExtendedPC os::Bsd::fetch_frame_from_ucontext(Thread* thread,
    ucontext_t* uc, intptr_t** ret_sp, intptr_t** ret_fp) {

    assert(thread != NULL, "just checking");
@@ -152,9 +331,9 @@
    ucontext_t* uc = (ucontext_t*)ucVoid;

    if (uc != NULL) {
-    epc = ExtendedPC(os::Linux::ucontext_get_pc(uc));
-    if (ret_sp) *ret_sp = os::Linux::ucontext_get_sp(uc);
-    if (ret_fp) *ret_fp = os::Linux::ucontext_get_fp(uc);
+    epc = ExtendedPC(os::Bsd::ucontext_get_pc(uc));
+    if (ret_sp) *ret_sp = os::Bsd::ucontext_get_sp(uc);
+    if (ret_fp) *ret_fp = os::Bsd::ucontext_get_fp(uc);
    } else {
        // construct empty ExtendedPC for return value checking
        epc = ExtendedPC(NULL);
@@ -217,7 +396,7 @@
    #endif // AMD64

    extern "C" JNIEXPORT int
-JVM_handle_linux_signal(int sig,
+JVM_handle_bsd_signal(int sig,
                        siginfo_t* info,
                        void* ucVoid,
                        int abort_if_unrecognized) {
@@ -230,13 +409,13 @@
    // Note: it's not uncommon that JNI code uses signal/sigset to install
    // then restore certain signal handler (e.g. to temporarily block SIGPIPE,
    // or have a SIGILL handler when detecting CPU type). When that happens,
-    // JVM_handle_linux_signal() might be invoked with junk info/ucVoid. To
+    // JVM_handle_bsd_signal() might be invoked with junk info/ucVoid. To
    // avoid unnecessary crash when libjsig is not preloaded, try handle signals
    // that do not require siginfo/ucontext first.

    if (sig == SIGPIPE || sig == SIGXFSZ) {
        // allow chained handler to go first
-    if (os::Linux::chained_handler(sig, info, ucVoid)) {
+    if (os::Bsd::chained_handler(sig, info, ucVoid)) {
        return true;
    } else {
        if (PrintMiscellaneous && (WizardMode || Verbose)) {
@@ -250,7 +429,7 @@

    JavaThread* thread = NULL;
    VMThread* vmthread = NULL;
-    if (os::Linux::signal_handlers_are_installed) {
+    if (os::Bsd::signal_handlers_are_installed) {
        if (t != NULL) {
            if (t->is_Java_thread()) {
                thread = (JavaThread*)t;
@@ -261,7 +440,7 @@
        }
    }
    /*
-    NOTE: does not seem to work on linux.
+    NOTE: does not seem to work on bsd.
    if (info == NULL || info->si_code <= 0 || info->si_code == SI_NOINFO) {
        // can't decode this kind of signal
        info = NULL;
@@ -276,21 +455,21 @@

    // %note os_trap_1
    if (info != NULL && uc != NULL && thread != NULL) {
-    pc = (address) os::Linux::ucontext_get_pc(uc);
+    pc = (address) os::Bsd::ucontext_get_pc(uc);

    if (pc == (address) Fetch32PFI) {
-    uc->uc_mcontext.gregs[REG_PC] = intptr_t(Fetch32Resume) ;
+    uc->context_pc = intptr_t(Fetch32Resume) ;
    }

```

```

        return 1 ;
    }
#ifdef AMD64
    if (pc == (address) FetchNPFI) {
-       uc->uc_mcontext.gregs[REG_PC] = intptr_t (FetchNResume) ;
+       uc->context_pc = intptr_t (FetchNResume) ;
        return 1 ;
    }
#endif // AMD64

    // Handle ALL stack overflow variations here
-   if (sig == SIGSEGV) {
+   if (sig == SIGSEGV || sig == SIGBUS) {
        address addr = (address) info->si_addr;

        // check if fault address is within thread stack
@@ -312,14 +491,15 @@
        // to handle_unexpected_exception way down below.
        thread->disable_stack_red_zone();
        tty->print_raw_cr("An irrecoverable stack overflow has occurred.");
#ifdef _ALLBSD_SOURCE
    } else {
        // Accessing stack address below sp may cause SEGV if current
        // thread has MAP_GROWSDOWN stack. This should only happen when
        // current thread was created by user code with MAP_GROWSDOWN flag
-       // and then attached to VM. See notes in os_linux.cpp.
+       // and then attached to VM. See notes in os_bsd.cpp.
        if (thread->osthread()->expanding_stack() == 0) {
            thread->osthread()->set_expanding_stack();
-           if (os::Linux::manually_expand_stack(thread, addr)) {
+           if (os::Bsd::manually_expand_stack(thread, addr)) {
                thread->osthread()->clear_expanding_stack();
                return 1;
            }
@@ -327,6 +507,7 @@
        } else {
            fatal("recursive segv. expanding stack.");
        }
    }
#endif
}
}
@@ -335,9 +516,16 @@
    // Java thread running in Java code => find exception handler if any
    // a fault inside compiled code, the interpreter, or a stub

-   if (sig == SIGSEGV && os::is_poll_address((address)info->si_addr)) {
+   if ((sig == SIGSEGV || sig == SIGBUS) && os::is_poll_address((address)info->si_addr)) {
        stub = SharedRuntime::get_poll_stub(pc);
#ifdef __APPLE__ && !defined(AMD64)
+   // 32-bit Darwin reports a SIGBUS for nearly all memory access exceptions.
+   // Catching SIGBUS here prevents the implicit SIGBUS NULL check below from
+   // being called, so only do so if the implicit NULL check is not necessary.
+   } else if (sig == SIGBUS && MacroAssembler::needs_explicit_null_check((int)info->si_addr)) {
#else
    } else if (sig == SIGBUS /* && info->si_code == BUS_OBJERR */) {
#endif
        // BugId 4454115: A read from a MappedByteBuffer can fault
        // here if the underlying file has been truncated.
        // Do not crash the VM in such a case.
@@ -358,9 +546,31 @@
                                pc,
                                SharedRuntime::
                                IMPLICIT_DIVIDE_BY_ZERO);
#ifdef __APPLE__
+   } else if (sig == SIGFPE && info->si_code == FPE_NOOP) {
+       int op = pc[0];
+
+       // Skip REX
+       if ((pc[0] & 0xf0) == 0x40) {
+           op = pc[1];

```

```

+         } else {
+             op = pc[0];
+         }
+
+         // Check for IDIV
+         if (op == 0xF7) {
+             stub = SharedRuntime::continuation_for_implicit_exception(thread, pc, SharedRuntime::
IMPLICIT_DIVIDE_BY_ZERO);
+         } else {
+             // TODO: handle more cases if we are using other x86 instructions
+             // that can generate SIGFPE signal.
+             tty->print_cr("unknown opcode 0x%X with SIGFPE.", op);
+             fatal("please update this code.");
+         }
+ #endif /* __APPLE__ */
+
+ #else
+     if (sig == SIGFPE /* && info->si_code == FPE_INTDIV */) {
-         // HACK: si_code does not work on linux 2.2.12-20!!!
+         // HACK: si_code does not work on bsd 2.2.12-20!!!
+         int op = pc[0];
+         if (op == 0xDB) {
+             // FIST
@@ -380,12 +590,12 @@
+             stub = SharedRuntime::continuation_for_implicit_exception(thread, pc, SharedRuntime::
IMPLICIT_DIVIDE_BY_ZERO);
+         } else {
+             // TODO: handle more cases if we are using other x86 instructions
-             // that can generate SIGFPE signal on linux.
+             // that can generate SIGFPE signal on bsd.
+             tty->print_cr("unknown opcode 0x%X with SIGFPE.", op);
+             fatal("please update this code.");
+         }
+ #endif // AMD64
-     } else if (sig == SIGSEGV &&
+     } else if ((sig == SIGSEGV || sig == SIGBUS) &&
+         !MacroAssembler::needs_explicit_null_check((intptr_t)info->si_addr)) {
+         // Determination of interpreter/vtable stub/compiled code null exception
+         stub = SharedRuntime::continuation_for_implicit_exception(thread, pc, SharedRuntime::IMPLICIT_NULL);
@@ -409,7 +619,7 @@
+         // process of write protecting the memory serialization page.
+         // It write enables the page immediately after protecting it
+         // so we can just return to retry the write.
-         if ((sig == SIGSEGV) &&
+         if ((sig == SIGSEGV || sig == SIGBUS) &&
+             os::is_memory_serialize_page(thread, (address) info->si_addr)) {
+             // Block current thread until the memory serialize page permission restored.
+             os::block_on_serialize_page_trap();
@@ -430,10 +640,10 @@
+         // Furthermore, a false-positive should be harmless.
+         if (UnguardOnExecutionViolation > 0 &&
+             (sig == SIGSEGV || sig == SIGBUS) &&
-             uc->uc_mcontext.gregs[REG_TRAPNO] == trap_page_fault) {
+             uc->context_trapno == trap_page_fault) {
+             int page_size = os::vm_page_size();
+             address addr = (address) info->si_addr;
-             address pc = os::Linux::ucontext_get_pc(uc);
+             address pc = os::Bsd::ucontext_get_pc(uc);
+             // Make sure the pc and the faulting address are sane.
+             //
+             // If an instruction spans a page boundary, and the page containing
@@ -500,12 +710,12 @@
+             // save all thread context in case we need to restore it
+             if (thread != NULL) thread->set_saved_exception_pc(pc);
-
-             uc->uc_mcontext.gregs[REG_PC] = (greg_t)stub;
+             uc->context_pc = (intptr_t)stub;
+             return true;
+         }
+     }
+
+ // signal-chaining

```

```

- if (os::Linux::chained_handler(sig, info, ucVoid)) {
+ if (os::Bsd::chained_handler(sig, info, ucVoid)) {
    return true;
}

@@ -515,7 +725,7 @@
}

    if (pc == NULL && uc != NULL) {
-    pc = os::Linux::ucontext_get_pc(uc);
+    pc = os::Bsd::ucontext_get_pc(uc);
    }

    // unmask current signal
@@ -530,14 +740,25 @@
    ShouldNotReachHere();
}

-void os::Linux::init_thread_fpu_state(void) {
+#ifdef _ALLBSD_SOURCE
+// From solaris_i486.s ported to bsd_i486.s
+extern "C" void fixcw();
+#endif
+
+void os::Bsd::init_thread_fpu_state(void) {
    #ifndef AMD64
+# ifdef _ALLBSD_SOURCE
+ // Set fpu to 53 bit precision. This happens too early to use a stub.
+ fixcw();
+# else
    // set fpu to 53 bit precision
    set_fpu_control_word(0x27f);
+# endif
    #endif // !AMD64
}

-int os::Linux::get_fpu_control_word(void) {
+#ifndef _ALLBSD_SOURCE
+int os::Bsd::get_fpu_control_word(void) {
    #ifdef AMD64
        return 0;
    #else
@@ -547,16 +768,17 @@
    #endif // AMD64
}

-void os::Linux::set_fpu_control_word(int fpu_control) {
+void os::Bsd::set_fpu_control_word(int fpu_control) {
    #ifndef AMD64
        _FPU_SETCW(fpu_control);
    #endif // !AMD64
}
+#endif

-// Check that the linux kernel version is 2.4 or higher since earlier
+// Check that the bsd kernel version is 2.4 or higher since earlier
// versions do not support SSE without patches.
bool os::supports_sse() {
-#ifdef AMD64
+#if defined(AMD64) || defined(_ALLBSD_SOURCE)
    return true;
    #else
        struct utsname uts;
@@ -599,34 +821,37 @@
    // thread stack

    #ifdef AMD64
- size_t os::Linux::min_stack_allowed = 64 * K;
+ size_t os::Bsd::min_stack_allowed = 64 * K;

    // amd64: pthread on amd64 is always in floating stack mode

```



```

-bool os::Linux::supports_variable_stack_size() { return true; }
+bool os::Bsd::supports_variable_stack_size() { return true; }
#else
-size_t os::Linux::min_stack_allowed = (48 DEBUG_ONLY(+4))*K;
+size_t os::Bsd::min_stack_allowed = (48 DEBUG_ONLY(+4))*K;

#ifdef __GNUC__
#define GET_GS() ((int gs; __asm__ volatile("movw %%gs, %w0":"=q"(gs)); gs&0xffff;})
#endif

-// Test if pthread library can support variable thread stack size. LinuxThreads
-// in fixed stack mode allocates 2M fixed slot for each thread. LinuxThreads
+#ifdef _ALLBSD_SOURCE
+bool os::Bsd::supports_variable_stack_size() { return true; }
+#else
+// Test if pthread library can support variable thread stack size. BsdThreads
+// in fixed stack mode allocates 2M fixed slot for each thread. BsdThreads
+// in floating stack mode and NPTL support variable stack size.
-bool os::Linux::supports_variable_stack_size() {
- if (os::Linux::is_NPTL()) {
+bool os::Bsd::supports_variable_stack_size() {
+ if (os::Bsd::is_NPTL()) {
+ // NPTL, yes
+ return true;

} else {
+ // Note: We can't control default stack size when creating a thread.
+ // If we use non-default stack size (pthread_attr_setstacksize), both
- // floating stack and non-floating stack LinuxThreads will return the
+ // floating stack and non-floating stack BsdThreads will return the
+ // same value. This makes it impossible to implement this function by
+ // detecting thread stack size directly.
+ //
- // An alternative approach is to check %gs. Fixed-stack LinuxThreads
- // do not use %gs, so its value is 0. Floating-stack LinuxThreads use
+ // An alternative approach is to check %gs. Fixed-stack BsdThreads
+ // do not use %gs, so its value is 0. Floating-stack BsdThreads use
+ // %gs (either as LDT selector or GDT selector, depending on kernel)
+ // to access thread specific data.
+ //
@@ -635,7 +860,7 @@
+ // Redhat confirmed that all known offenders have been modified to use
+ // either %fs or TSD). In the worst case scenario, when VM is embedded in
+ // a native application that plays with %gs, we might see non-zero %gs
- // even LinuxThreads is running in fixed stack mode. As the result, we'll
+ // even BsdThreads is running in fixed stack mode. As the result, we'll
+ // return true and skip _thread_safety_check(), so we may not be able to
+ // detect stack-heap collisions. But otherwise it's harmless.
+ //
@@ -646,10 +871,11 @@
+ #endif
+ }
+ #endif
+ #endif // AMD64

+ // return default stack size for thr_type
-size_t os::Linux::default_stack_size(os::ThreadType thr_type) {
+size_t os::Bsd::default_stack_size(os::ThreadType thr_type) {
+ // default stack size (compiler thread needs larger stack)
+ #ifdef AMD64
+ size_t s = (thr_type == os::compiler_thread ? 4 * M : 1 * M);
@@ -659,7 +885,7 @@
+ return s;
+ }

-size_t os::Linux::default_guard_size(os::ThreadType thr_type) {
+size_t os::Bsd::default_guard_size(os::ThreadType thr_type) {
+ // Creating guard page is very expensive. Java thread has HotSpot
+ // guard page, only enable glibc guard page for non-Java threads.
+ return (thr_type == java_thread ? 0 : page_size());

```

```

@@ -699,11 +925,46 @@
// pthread_attr_getstack()

static void current_stack_region(address * bottom, size_t * size) {
- if (os::Linux::is_initial_thread()) {
+#ifdef __APPLE__
+ pthread_t self = pthread_self();
+ void *stacktop = pthread_get_stackaddr_np(self);
+ *size = pthread_get_stacksize_np(self);
+ *bottom = (address) stacktop - *size;
+#elif defined(__OpenBSD__)
+ stack_t ss;
+ int rslt = pthread_stackseg_np(pthread_self(), &ss);
+
+ if (rslt != 0)
+ fatal(err_msg("pthread_stackseg_np failed with err = %d", rslt));
+
+ *bottom = (address)((char *)ss.ss_sp - ss.ss_size);
+ *size = ss.ss_size;
+#elif defined(_ALLBSD_SOURCE)
+ pthread_attr_t attr;
+
+ int rslt = pthread_attr_init(&attr);
+
+ // JVM needs to know exact stack location, abort if it fails
+ if (rslt != 0)
+ fatal(err_msg("pthread_attr_init failed with err = %d", rslt));
+
+ rslt = pthread_attr_get_np(pthread_self(), &attr);
+
+ if (rslt != 0)
+ fatal(err_msg("pthread_attr_get_np failed with err = %d", rslt));
+
+ if (pthread_attr_getstackaddr(&attr, (void **)bottom) != 0 ||
+ pthread_attr_getstacksize(&attr, size) != 0) {
+ fatal("Can not locate current stack attributes!");
+ }
+
+ pthread_attr_destroy(&attr);
+#else
+ if (os::Bsd::is_initial_thread()) {
+ // initial thread needs special handling because pthread_getattr_np()
+ // may return bogus value.
- *bottom = os::Linux::initial_thread_stack_bottom();
- *size = os::Linux::initial_thread_stack_size();
+ *bottom = os::Bsd::initial_thread_stack_bottom();
+ *size = os::Bsd::initial_thread_stack_size();
+ } else {
+ pthread_attr_t attr;

@@ -725,6 +986,7 @@
pthread_attr_destroy(&attr);

}
+#endif
assert(os::current_stack_pointer() >= *bottom &&
os::current_stack_pointer() < *bottom + *size, "just checking");
}
@@ -753,51 +1015,49 @@
ucontext_t *uc = (ucontext_t*)context;
st->print_cr("Registers:");
#ifdef AMD64
- st->print(" RAX=" INTPTR_FORMAT, uc->uc_mcontext.gregs[REG_RAX]);
- st->print(" RBX=" INTPTR_FORMAT, uc->uc_mcontext.gregs[REG_RBX]);
- st->print(" RCX=" INTPTR_FORMAT, uc->uc_mcontext.gregs[REG_RCX]);
- st->print(" RDX=" INTPTR_FORMAT, uc->uc_mcontext.gregs[REG_RDX]);
+ st->print(" RAX=" INTPTR_FORMAT, uc->context_rax);
+ st->print(" RBX=" INTPTR_FORMAT, uc->context_rbx);
+ st->print(" RCX=" INTPTR_FORMAT, uc->context_rcx);
+ st->print(" RDX=" INTPTR_FORMAT, uc->context_rdx);
st->cr();

```

```

- st->print( "RSP=" INTPTR_FORMAT, uc->uc_mcontext.gregs[REG_RSP]);
- st->print( " RBP=" INTPTR_FORMAT, uc->uc_mcontext.gregs[REG_RBP]);
- st->print( " RSI=" INTPTR_FORMAT, uc->uc_mcontext.gregs[REG_RSI]);
- st->print( " RDI=" INTPTR_FORMAT, uc->uc_mcontext.gregs[REG_RDI]);
+ st->print( "RSP=" INTPTR_FORMAT, uc->context_rsp);
+ st->print( " RBP=" INTPTR_FORMAT, uc->context_rbp);
+ st->print( " RSI=" INTPTR_FORMAT, uc->context_rsi);
+ st->print( " RDI=" INTPTR_FORMAT, uc->context_rdi);
  st->cr();
- st->print( "R8 =" INTPTR_FORMAT, uc->uc_mcontext.gregs[REG_R8]);
- st->print( " R9 =" INTPTR_FORMAT, uc->uc_mcontext.gregs[REG_R9]);
- st->print( " R10=" INTPTR_FORMAT, uc->uc_mcontext.gregs[REG_R10]);
- st->print( " R11=" INTPTR_FORMAT, uc->uc_mcontext.gregs[REG_R11]);
+ st->print( "R8 =" INTPTR_FORMAT, uc->context_r8);
+ st->print( " R9 =" INTPTR_FORMAT, uc->context_r9);
+ st->print( " R10=" INTPTR_FORMAT, uc->context_r10);
+ st->print( " R11=" INTPTR_FORMAT, uc->context_r11);
  st->cr();
- st->print( "R12=" INTPTR_FORMAT, uc->uc_mcontext.gregs[REG_R12]);
- st->print( " R13=" INTPTR_FORMAT, uc->uc_mcontext.gregs[REG_R13]);
- st->print( " R14=" INTPTR_FORMAT, uc->uc_mcontext.gregs[REG_R14]);
- st->print( " R15=" INTPTR_FORMAT, uc->uc_mcontext.gregs[REG_R15]);
+ st->print( "R12=" INTPTR_FORMAT, uc->context_r12);
+ st->print( " R13=" INTPTR_FORMAT, uc->context_r13);
+ st->print( " R14=" INTPTR_FORMAT, uc->context_r14);
+ st->print( " R15=" INTPTR_FORMAT, uc->context_r15);
  st->cr();
- st->print( "RIP=" INTPTR_FORMAT, uc->uc_mcontext.gregs[REG_RIP]);
- st->print( " EFLAGS=" INTPTR_FORMAT, uc->uc_mcontext.gregs[REG_EFL]);
- st->print( " CSGSFS=" INTPTR_FORMAT, uc->uc_mcontext.gregs[REG_CSGSFS]);
- st->print( " ERR=" INTPTR_FORMAT, uc->uc_mcontext.gregs[REG_ERR]);
+ st->print( "RIP=" INTPTR_FORMAT, uc->context_rip);
+ st->print( " EFLAGS=" INTPTR_FORMAT, uc->context_flags);
+ st->print( " ERR=" INTPTR_FORMAT, uc->context_err);
  st->cr();
- st->print( " TRAPNO=" INTPTR_FORMAT, uc->uc_mcontext.gregs[REG_TRAPNO]);
+ st->print( " TRAPNO=" INTPTR_FORMAT, uc->context_trapno);
#else
- st->print( "EAX=" INTPTR_FORMAT, uc->uc_mcontext.gregs[REG_EAX]);
- st->print( " EBX=" INTPTR_FORMAT, uc->uc_mcontext.gregs[REG_EBX]);
- st->print( " ECX=" INTPTR_FORMAT, uc->uc_mcontext.gregs[REG_ECX]);
- st->print( " EDX=" INTPTR_FORMAT, uc->uc_mcontext.gregs[REG_EDX]);
+ st->print( "EAX=" INTPTR_FORMAT, uc->context_eax);
+ st->print( " EBX=" INTPTR_FORMAT, uc->context_ebx);
+ st->print( " ECX=" INTPTR_FORMAT, uc->context_ecx);
+ st->print( " EDX=" INTPTR_FORMAT, uc->context_edx);
  st->cr();
- st->print( "ESP=" INTPTR_FORMAT, uc->uc_mcontext.gregs[REG_UESP]);
- st->print( " EBP=" INTPTR_FORMAT, uc->uc_mcontext.gregs[REG_EBP]);
- st->print( " ESI=" INTPTR_FORMAT, uc->uc_mcontext.gregs[REG_ESI]);
- st->print( " EDI=" INTPTR_FORMAT, uc->uc_mcontext.gregs[REG_EDI]);
+ st->print( "ESP=" INTPTR_FORMAT, uc->context_esp);
+ st->print( " EBP=" INTPTR_FORMAT, uc->context_ebp);
+ st->print( " ESI=" INTPTR_FORMAT, uc->context_esi);
+ st->print( " EDI=" INTPTR_FORMAT, uc->context_edi);
  st->cr();
- st->print( "EIP=" INTPTR_FORMAT, uc->uc_mcontext.gregs[REG_EIP]);
- st->print( " EFLAGS=" INTPTR_FORMAT, uc->uc_mcontext.gregs[REG_EFL]);
- st->print( " CR2=" INTPTR_FORMAT, uc->uc_mcontext.cr2);
+ st->print( "EIP=" INTPTR_FORMAT, uc->context_eip);
+ st->print( " EFLAGS=" INTPTR_FORMAT, uc->context_eflags);
#endif // AMD64
  st->cr();
  st->cr();

- intptr_t *sp = (intptr_t *)os::Linux::ucontext_get_sp(uc);
+ intptr_t *sp = (intptr_t *)os::Bsd::ucontext_get_sp(uc);
  st->print_cr("Top of Stack: (sp=" PTR_FORMAT ")", sp);
  print_hex_dump(st, (address)sp, (address)(sp + 8*sizeof(intptr_t)), sizeof(intptr_t));
  st->cr();
@@ -805,7 +1065,7 @@

```

```

// Note: it may be unsafe to inspect memory near pc. For example, pc may
// point to garbage if entry point in an mmethod is corrupted. Leave
// this at the end, and hope for the best.
- address pc = os::Linux::ucontext_get_pc(uc);
+ address pc = os::Bsd::ucontext_get_pc(uc);
st->print_cr("Instructions: (pc=" PTR_FORMAT ")", pc);
print_hex_dump(st, pc - 32, pc + 32, sizeof(char));
}
@@ -825,31 +1085,31 @@
// this is only for the "general purpose" registers

#ifdef AMD64
- st->print("RAX="); print_location(st, uc->uc_mcontext.gregs[REG_RAX]);
- st->print("RBX="); print_location(st, uc->uc_mcontext.gregs[REG_RBX]);
- st->print("RCX="); print_location(st, uc->uc_mcontext.gregs[REG_RCX]);
- st->print("RDX="); print_location(st, uc->uc_mcontext.gregs[REG_RDX]);
- st->print("RSP="); print_location(st, uc->uc_mcontext.gregs[REG_RSP]);
- st->print("RBP="); print_location(st, uc->uc_mcontext.gregs[REG_RBP]);
- st->print("RSI="); print_location(st, uc->uc_mcontext.gregs[REG_RSI]);
- st->print("RDI="); print_location(st, uc->uc_mcontext.gregs[REG_RDI]);
- st->print("R8 ="); print_location(st, uc->uc_mcontext.gregs[REG_R8]);
- st->print("R9 ="); print_location(st, uc->uc_mcontext.gregs[REG_R9]);
- st->print("R10="); print_location(st, uc->uc_mcontext.gregs[REG_R10]);
- st->print("R11="); print_location(st, uc->uc_mcontext.gregs[REG_R11]);
- st->print("R12="); print_location(st, uc->uc_mcontext.gregs[REG_R12]);
- st->print("R13="); print_location(st, uc->uc_mcontext.gregs[REG_R13]);
- st->print("R14="); print_location(st, uc->uc_mcontext.gregs[REG_R14]);
- st->print("R15="); print_location(st, uc->uc_mcontext.gregs[REG_R15]);
-#else
- st->print("EAX="); print_location(st, uc->uc_mcontext.gregs[REG_EAX]);
- st->print("EBX="); print_location(st, uc->uc_mcontext.gregs[REG_EBX]);
- st->print("ECX="); print_location(st, uc->uc_mcontext.gregs[REG_ECX]);
- st->print("EDX="); print_location(st, uc->uc_mcontext.gregs[REG_EDX]);
- st->print("ESP="); print_location(st, uc->uc_mcontext.gregs[REG_ESP]);
- st->print("EBP="); print_location(st, uc->uc_mcontext.gregs[REG_EBP]);
- st->print("ESI="); print_location(st, uc->uc_mcontext.gregs[REG_ESI]);
- st->print("EDI="); print_location(st, uc->uc_mcontext.gregs[REG_EDI]);
+ st->print("RAX="); print_location(st, uc->context_rax);
+ st->print("RBX="); print_location(st, uc->context_rbx);
+ st->print("RCX="); print_location(st, uc->context_rcx);
+ st->print("RDX="); print_location(st, uc->context_rdx);
+ st->print("RSP="); print_location(st, uc->context_rsp);
+ st->print("RBP="); print_location(st, uc->context_rbp);
+ st->print("RSI="); print_location(st, uc->context_rsi);
+ st->print("RDI="); print_location(st, uc->context_rdi);
+ st->print("R8 ="); print_location(st, uc->context_r8);
+ st->print("R9 ="); print_location(st, uc->context_r9);
+ st->print("R10="); print_location(st, uc->context_r10);
+ st->print("R11="); print_location(st, uc->context_r11);
+ st->print("R12="); print_location(st, uc->context_r12);
+ st->print("R13="); print_location(st, uc->context_r13);
+ st->print("R14="); print_location(st, uc->context_r14);
+ st->print("R15="); print_location(st, uc->context_r15);
+#else
+ st->print("EAX="); print_location(st, uc->context_eax);
+ st->print("EBX="); print_location(st, uc->context_ebx);
+ st->print("ECX="); print_location(st, uc->context_ecx);
+ st->print("EDX="); print_location(st, uc->context_edx);
+ st->print("ESP="); print_location(st, uc->context_esp);
+ st->print("EBP="); print_location(st, uc->context_ebp);
+ st->print("ESI="); print_location(st, uc->context_esi);
+ st->print("EDI="); print_location(st, uc->context_edi);
#endif // AMD64

st->cr();
--- src/os_cpu/linux_x86/vm/os_linux_x86.hpp 2011-07-26 20:21:21.000000000 -0600
+++ src/os_cpu/bsd_x86/vm/os_bsd_x86.hpp 2011-07-26 20:21:21.000000000 -0600
@@ -22,8 +22,8 @@
*
*/

```

```

-#ifndef OS_CPU_LINUX_X86_VM_OS_LINUX_X86_HPP
-#define OS_CPU_LINUX_X86_VM_OS_LINUX_X86_HPP
+#ifndef OS_CPU_BSD_X86_VM_OS_BSD_X86_HPP
+#define OS_CPU_BSD_X86_VM_OS_BSD_X86_HPP

    static void setup_fpu();
    static bool supports_sse();
@@ -34,4 +34,4 @@
    // Note: Currently only used in 64 bit Windows implementations
    static bool register_code_area(char *low, char *high) { return true; }

-#endif // OS_CPU_LINUX_X86_VM_OS_LINUX_X86_HPP
+#endif // OS_CPU_BSD_X86_VM_OS_BSD_X86_HPP
--- src/os_cpu/linux_x86/vm/prefetch_linux_x86.inline.hpp      2011-07-26 20:21:21.000000000 -0600
+++ src/os_cpu/bsd_x86/vm/prefetch_bsd_x86.inline.hpp         2011-07-26 20:21:21.000000000 -0600
@@ -22,8 +22,8 @@
 *
 */

-#ifndef OS_CPU_LINUX_X86_VM_PREFETCH_LINUX_X86_INLINE_HPP
-#define OS_CPU_LINUX_X86_VM_PREFETCH_LINUX_X86_INLINE_HPP
+#ifndef OS_CPU_BSD_X86_VM_PREFETCH_BSD_X86_INLINE_HPP
+#define OS_CPU_BSD_X86_VM_PREFETCH_BSD_X86_INLINE_HPP

#include "runtime/prefetch.hpp"

@@ -44,4 +44,4 @@
#endif // AMD64
}

-#endif // OS_CPU_LINUX_X86_VM_PREFETCH_LINUX_X86_INLINE_HPP
+#endif // OS_CPU_BSD_X86_VM_PREFETCH_BSD_X86_INLINE_HPP
--- src/os_cpu/linux_x86/vm/threadLS_linux_x86.cpp           2011-07-26 20:21:21.000000000 -0600
+++ src/os_cpu/bsd_x86/vm/threadLS_bsd_x86.cpp              2011-07-26 20:21:21.000000000 -0600
@@ -24,7 +24,7 @@

#include "precompiled.hpp"
#include "runtime/threadLocalStorage.hpp"
-#include "thread_linux.inline.hpp"
+#include "thread_bsd.inline.hpp"

// Map stack pointer (%esp) to thread pointer for faster TLS access
//
@@ -41,7 +41,7 @@
//
// If an application creates and destroys a lot of threads, usually the
// stack space freed by a thread will soon get reused by new thread
-// (this is especially true in NPTL or LinuxThreads in fixed-stack mode).
+// (this is especially true in NPTL or BsdThreads in fixed-stack mode).
// No memory page in _sp_map is wasted.
//
// However, it's still possible that we might end up populating &
@@ -49,7 +49,7 @@
// amount of live data in the table could be quite small. The max wastage
// is less than 4M bytes. If it becomes an issue, we could use madvise()
// with MADV_DONTNEED to reclaim unused (i.e. all-zero) pages in _sp_map.
-// MADV_DONTNEED on Linux keeps the virtual memory mapping, but zaps the
+// MADV_DONTNEED on Bsd keeps the virtual memory mapping, but zaps the
// physical memory page (i.e. similar to MADV_FREE on Solaris).

#ifndef AMD64
--- src/os_cpu/linux_x86/vm/threadLS_linux_x86.hpp           2011-07-26 20:21:21.000000000 -0600
+++ src/os_cpu/bsd_x86/vm/threadLS_bsd_x86.hpp              2011-07-26 20:21:21.000000000 -0600
@@ -22,16 +22,18 @@
 *
 */

-#ifndef OS_CPU_LINUX_X86_VM_THREADLS_LINUX_X86_HPP
-#define OS_CPU_LINUX_X86_VM_THREADLS_LINUX_X86_HPP
+#ifndef OS_CPU_BSD_X86_VM_THREADLS_BSD_X86_HPP
+#define OS_CPU_BSD_X86_VM_THREADLS_BSD_X86_HPP

```

```

// Processor dependent parts of ThreadLocalStorage

#ifdef AMD64
- // map stack pointer to thread pointer - see notes in threadLS_linux_x86.cpp
+ // map stack pointer to thread pointer - see notes in threadLS_bsd_x86.cpp
#define SP_BITLENGTH 32
#ifdef PAGE_SHIFT
#define PAGE_SHIFT 12
#define PAGE_SIZE (1UL << PAGE_SHIFT)
#endif
static Thread* _sp_map[1UL << (SP_BITLENGTH - PAGE_SHIFT)];
#endif // !AMD64

@@ -51,4 +53,4 @@
#endif // AMD64
}

-#endif // OS_CPU_LINUX_X86_VM_THREADLS_LINUX_X86_HPP
+#endif // OS_CPU_BSD_X86_VM_THREADLS_BSD_X86_HPP
--- src/os_cpu/linux_x86/vm/thread_linux_x86.cpp 2011-07-26 20:21:21.000000000 -0600
+++ src/os_cpu/bsd_x86/vm/thread_bsd_x86.cpp 2011-07-26 20:21:21.000000000 -0600
@@ -24,7 +24,7 @@

#include "precompiled.hpp"
#include "runtime/frame.inline.hpp"
-#include "thread_linux.inline.hpp"
+#include "thread_bsd.inline.hpp"

// For Forte Analyzer AsyncGetCallTrace profiling support - thread is
// currently interrupted by SIGPROF
@@ -51,7 +51,7 @@

intptr_t* ret_fp;
intptr_t* ret_sp;
- ExtendedPC addr = os::Linux::fetch_frame_from_ucontext(this, uc,
+ ExtendedPC addr = os::Bsd::fetch_frame_from_ucontext(this, uc,
&ret_sp, &ret_fp);
if (addr.pc() == NULL || ret_sp == NULL) {
// ucontext wasn't useful
--- src/os_cpu/linux_x86/vm/thread_linux_x86.hpp 2011-07-26 20:21:21.000000000 -0600
+++ src/os_cpu/bsd_x86/vm/thread_bsd_x86.hpp 2011-07-26 20:21:21.000000000 -0600
@@ -22,8 +22,8 @@
*
*/

-#ifndef OS_CPU_LINUX_X86_VM_THREAD_LINUX_X86_HPP
-#define OS_CPU_LINUX_X86_VM_THREAD_LINUX_X86_HPP
+#ifndef OS_CPU_BSD_X86_VM_THREAD_BSD_X86_HPP
+#define OS_CPU_BSD_X86_VM_THREAD_BSD_X86_HPP

private:
void pd_initialize() {
@@ -67,4 +67,4 @@
static void enable_register_stack_guard() {}
static void disable_register_stack_guard() {}

-#endif // OS_CPU_LINUX_X86_VM_THREAD_LINUX_X86_HPP
+#endif // OS_CPU_BSD_X86_VM_THREAD_BSD_X86_HPP
--- src/os_cpu/linux_x86/vm/vmStructs_linux_x86.hpp 2011-07-26 20:21:21.000000000 -0600
+++ src/os_cpu/bsd_x86/vm/vmStructs_bsd_x86.hpp 2011-07-26 20:21:21.000000000 -0600
@@ -22,8 +22,8 @@
*
*/

-#ifndef OS_CPU_LINUX_X86_VM_VMSTRUCTS_LINUX_X86_HPP
-#define OS_CPU_LINUX_X86_VM_VMSTRUCTS_LINUX_X86_HPP
+#ifndef OS_CPU_BSD_X86_VM_VMSTRUCTS_BSD_X86_HPP
+#define OS_CPU_BSD_X86_VM_VMSTRUCTS_BSD_X86_HPP

// These are the OS and CPU-specific fields, types and integer

```

```

// constants required by the Serviceability Agent. This file is
@@ -34,7 +34,7 @@

/*****
\
/* Threads (NOTE: incomplete)
*/

/*****
\
- nonstatic_field(OSThread,          _thread_id,
pid_t)
+ nonstatic_field(OSThread,          _thread_id,
pthread_t)
  nonstatic_field(OSThread,          _pthread_id,
pthread_t)
/* This must be the last entry, and must be present
*/
last_entry()
@@ -62,4 +62,4 @@
/* This must be the last entry, and must be present */
last_entry()

-#endif // OS_CPU_LINUX_X86_VM_VMSTRUCTS_LINUX_X86_HPP
+#endif // OS_CPU_BSD_X86_VM_VMSTRUCTS_BSD_X86_HPP
Files src/os_cpu/linux_x86/vm/vm_version_linux_x86.cpp and src/os_cpu/bsd_x86/vm/vm_version_bsd_x86.cpp are
identical
Files src/os_cpu/linux_zero/vm/assembly_linux_zero.cpp and src/os_cpu/bsd_zero/vm/assembly_bsd_zero.cpp are
identical
--- src/os_cpu/linux_zero/vm/atomic_linux_zero.inline.hpp          2011-07-26 20:21:21.000000000 -0600
+++ src/os_cpu/bsd_zero/vm/atomic_bsd_zero.inline.hpp              2011-07-26 20:21:21.000000000 -0600
@@ -23,144 +23,463 @@
*
*/

-#ifndef OS_CPU_LINUX_ZERO_VM_ATOMIC_LINUX_ZERO_INLINE_HPP
-#define OS_CPU_LINUX_ZERO_VM_ATOMIC_LINUX_ZERO_INLINE_HPP
+#ifndef OS_CPU_BSD_ZERO_VM_ATOMIC_BSD_ZERO_INLINE_HPP
+#define OS_CPU_BSD_ZERO_VM_ATOMIC_BSD_ZERO_INLINE_HPP

-#include "orderAccess_linux_zero.inline.hpp"
+#include "orderAccess_bsd_zero.inline.hpp"
#include "runtime/atomic.hpp"
#include "runtime/os.hpp"
#include "vm_version_zero.hpp"

-// Implementation of class atomic
-
-#ifdef M68K
+#include <sys/types.h>
+#ifdef __NetBSD__
+#include <sys/atomic.h>
+#elif __FreeBSD__
+
+#include <sys/types.h>
+#ifndef SPARC
+#include <machine/atomic.h>
+#else
+
+/*
+ * __m68k_cmpxchg
+ * On FreeBSD/sparc64, <machine/atomic.h> pulls in <machine/cpufunc.h>
+ * which includes definitions which cause conflicts with various
+ * definitions within HotSpot source. To avoid that, pull in those
+ * definitions verbatim instead of including the header. Yuck.
+ */
+
+/*-
+ * Copyright (c) 1998 Doug Rabson.
+ * Copyright (c) 2001 Jake Burkholder.
+ * All rights reserved.

```

```

*
- * Atomically store newval in *ptr if *ptr is equal to oldval for user space.
- * Returns newval on success and oldval if no exchange happened.
- * This implementation is processor specific and works on
- * 68020 68030 68040 and 68060.
+ * Redistribution and use in source and binary forms, with or without
+ * modification, are permitted provided that the following conditions
+ * are met:
+ * 1. Redistributions of source code must retain the above copyright
+ * notice, this list of conditions and the following disclaimer.
+ * 2. Redistributions in binary form must reproduce the above copyright
+ * notice, this list of conditions and the following disclaimer in the
+ * documentation and/or other materials provided with the distribution.
*
- * It will not work on ColdFire, 68000 and 68010 since they lack the CAS
- * instruction.
- * Using a kernelhelper would be better for arch complete implementation.
+ * THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND
+ * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
+ * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
+ * ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
+ * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
+ * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
+ * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
+ * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
+ * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
+ * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
+ * SUCH DAMAGE.
*
*/

-static inline int __m68k_cmpxchg(int oldval, int newval, volatile int *ptr) {
- int ret;
- __asm __volatile ("cas%.l %0,%2,%1"
- : "=d" (ret), "+m" (*(ptr))
- : "d" (newval), "0" (oldval));
- return ret;
-}
-
-/* Perform an atomic compare and swap: if the current value of `*PTR'
- is OLDVAL, then write NEWVAL into `*PTR'. Return the contents of
- `*PTR' before the operation.*/
-static inline int m68k_compare_and_swap(volatile int *ptr,
- int oldval,
- int newval) {
- for (;;) {
- int prev = *ptr;
- if (prev != oldval)
- return prev;
-
- if (__m68k_cmpxchg (prev, newval, ptr) == newval)
- // Success.
- return prev;
-
- // We failed even though prev == oldval. Try again.
- }
-}
-
-/* Atomically add an int to memory. */
-static inline int m68k_add_and_fetch(volatile int *ptr, int add_value) {
- for (;;) {
- // Loop until success.
-
- int prev = *ptr;
-
- if (__m68k_cmpxchg (prev, prev + add_value, ptr) == prev + add_value)
- return prev + add_value;
- }
-}
-
-/* Atomically write VALUE into `*PTR' and returns the previous

```



```

- contents of `*PTR'. */
-static inline int m68k_lock_test_and_set(volatile int *ptr, int newval) {
- for (;;) {
-     // Loop until success.
-     int prev = *ptr;
-
-     if (__m68k_cmpxchg (prev, newval, ptr) == prev)
-         return prev;
- }
-}
-#endif // M68K
+#include <machine/asi.h>

-#ifndef ARM
+/*
+ * Membar operand macros for use in other macros when # is a special
+ * character.  Keep these in sync with what the hardware expects.
+ */
+#define M_LoadLoad (0)
+#define M_StoreLoad (1)
+#define M_LoadStore (2)
+#define M_StoreStore (3)
+
+#define CMASK_SHIFT (4)
+#define MMASK_SHIFT (0)
+
+#define CMASK_GEN(bit) ((1 << (bit)) << CMASK_SHIFT)
+#define MMASK_GEN(bit) ((1 << (bit)) << MMASK_SHIFT)
+
+#define LoadLoad MMASK_GEN(M_LoadLoad)
+#define StoreLoad MMASK_GEN(M_StoreLoad)
+#define LoadStore MMASK_GEN(M_LoadStore)
+#define StoreStore MMASK_GEN(M_StoreStore)
+
+#define casa(rs1, rs2, rd, asi) ({
+ u_int __rd = (uint32_t)(rd);
+ __asm __volatile("casa [%2] %3, %4, %0"
+ : "+r" (__rd), "=m" (*rs1)
+ : "r" (rs1), "n" (asi), "r" (rs2), "m" (*rs1));
+ __rd;
+})
+
+#define casxa(rs1, rs2, rd, asi) ({
+ u_long __rd = (uint64_t)(rd);
+ __asm __volatile("casxa [%2] %3, %4, %0"
+ : "+r" (__rd), "=m" (*rs1)
+ : "r" (rs1), "n" (asi), "r" (rs2), "m" (*rs1));
+ __rd;
+})
+
+#define membar(mask) do {
+ __asm __volatile("membar %0" : : "n" (mask) : "memory");
+} while (0)
+
+#ifdef _KERNEL
+#define __ASI_ATOMIC ASI_N
+#else
+#define __ASI_ATOMIC ASI_P
+#endif
+
+#define mb() __asm__ __volatile__ ("membar #MemIssue": : "memory")
+#define wmb() mb()
+#define rmb() mb()

/*
- * __kernel_cmpxchg
+ * Various simple arithmetic on memory which is atomic in the presence
+ * of interrupts and multiple processors. See atomic(9) for details.
+ * Note that efficient hardware support exists only for the 32 and 64
+ * bit variants; the 8 and 16 bit versions are not provided and should
+ * not be used in MI code.

```

```

 *
- * Atomically store newval in *ptr if *ptr is equal to oldval for user space.
- * Return zero if *ptr was changed or non-zero if no exchange happened.
- * The C flag is also set if *ptr was changed to allow for assembly
- * optimization in the calling code.
+ * This implementation takes advantage of the fact that the sparc64
+ * cas instruction is both a load and a store. The loop is often coded
+ * as follows:
 *
+ *      do {
+ *          expect = *p;
+ *          new = expect + 1;
+ *      } while (cas(p, expect, new) != expect);
+ *
+ * which performs an unnecessary load on each iteration that the cas
+ * operation fails. Modified as follows:
+ *
+ *      expect = *p;
+ *      for (;;) {
+ *          new = expect + 1;
+ *          result = cas(p, expect, new);
+ *          if (result == expect)
+ *              break;
+ *          expect = result;
+ *      }
+ *
+ * the return value of cas is used to avoid the extra reload.
+ *
+ * The memory barriers provided by the acq and rel variants are intended
+ * to be sufficient for use of relaxed memory ordering. Due to the
+ * suggested assembly syntax of the membar operands containing a #
+ * character, they cannot be used in macros. The cmask and mmask bits
+ * are hard coded in machine/cpufunc.h and used here through macros.
+ * Hopefully sun will choose not to change the bit numbers.
 */

-typedef int (__kernel_cmpxchg_t)(int oldval, int newval, volatile int *ptr);
-#define __kernel_cmpxchg (*(__kernel_cmpxchg_t *) 0xffff0fc0)
+#define      itype(sz)      uint ## sz ## _t

+#define      atomic_cas_32(p, e, s)      casa(p, e, s, __ASI_ATOMIC)
+#define      atomic_cas_64(p, e, s)      casxa(p, e, s, __ASI_ATOMIC)

+#define      atomic_cas(p, e, s, sz)
+      atomic_cas_ ## sz(p, e, s)

-/* Perform an atomic compare and swap: if the current value of `PTR'
- is OLDVAL, then write NEWVAL into `PTR'. Return the contents of
- `PTR' before the operation.*/
-static inline int arm_compare_and_swap(volatile int *ptr,
-                                       int oldval,
-                                       int newval) {
-    for (;;) {
-        int prev = *ptr;
-        if (prev != oldval)
-            return prev;
+#define      atomic_cas_acq(p, e, s, sz) ({
+    itype(sz) v;
+    v = atomic_cas(p, e, s, sz);
+    membar(LoadLoad | LoadStore);
+    v;
+})
+
+#define      atomic_cas_rel(p, e, s, sz) ({
+    itype(sz) v;
+    membar(LoadStore | StoreStore);
+    v = atomic_cas(p, e, s, sz);
+    v;
+})
+
+#define      atomic_op(p, op, v, sz) ({

```

```

+     itype(sz) e, r, s;
+     for (e = *(volatile itype(sz) *)p; e = r) {
+         s = e op v;
+         r = atomic_cas_ ## sz(p, e, s);
+         if (r == e)
+             break;
+     }
+     e;
+ })
+
+ #define atomic_op_acq(p, op, v, sz) ({
+     itype(sz) t;
+     t = atomic_op(p, op, v, sz);
+     membar(LoadLoad | LoadStore);
+     t;
+ })
+
+ #define atomic_op_rel(p, op, v, sz) ({
+     itype(sz) t;
+     membar(LoadStore | StoreStore);
+     t = atomic_op(p, op, v, sz);
+     t;
+ })
+
+ #define atomic_load(p, sz)
+     atomic_cas(p, 0, 0, sz)
+
+ #define atomic_load_acq(p, sz) ({
+     itype(sz) v;
+     v = atomic_load(p, sz);
+     membar(LoadLoad | LoadStore);
+     v;
+ })
+
+ #define atomic_load_clear(p, sz) ({
+     itype(sz) e, r;
+     for (e = *(volatile itype(sz) *)p; e = r) {
+         r = atomic_cas(p, e, 0, sz);
+         if (r == e)
+             break;
+     }
+     e;
+ })
+
+ #define atomic_store(p, v, sz) do {
+     itype(sz) e, r;
+     for (e = *(volatile itype(sz) *)p; e = r) {
+         r = atomic_cas(p, e, v, sz);
+         if (r == e)
+             break;
+     }
+ } while (0)
+
+ #define atomic_store_rel(p, v, sz) do {
+     membar(LoadStore | StoreStore);
+     atomic_store(p, v, sz);
+ } while (0)
+
+ #define ATOMIC_GEN(name, ptype, vtype, atype, sz)
+
+ #static __inline vtype
+ #atomic_add_ ## name(volatile ptype p, atype v)
+ {
+     return ((vtype)atomic_op(p, +, v, sz));
+ }
+
+ #static __inline vtype
+ #atomic_add_acq_ ## name(volatile ptype p, atype v)
+ {
+     return ((vtype)atomic_op_acq(p, +, v, sz));
+ }
+
+ #static __inline vtype

```

```

+atomic_add_rel_ ## name(volatile ptype p, atype v)
+{
+    return ((vtype)atomic_op_rel(p, +, v, sz));
+}
+
+static __inline int
+atomic_cmpset_ ## name(volatile ptype p, vtype e, vtype s)
+{
+    return (((vtype)atomic_cas(p, e, s, sz)) == e);
+}
+static __inline int
+atomic_cmpset_acq_ ## name(volatile ptype p, vtype e, vtype s)
+{
+    return (((vtype)atomic_cas_acq(p, e, s, sz)) == e);
+}
+static __inline int
+atomic_cmpset_rel_ ## name(volatile ptype p, vtype e, vtype s)
+{
+    return (((vtype)atomic_cas_rel(p, e, s, sz)) == e);
+}
+
+static __inline vtype
+atomic_load_ ## name(volatile ptype p)
+{
+    return ((vtype)atomic_cas(p, 0, 0, sz));
+}
+static __inline vtype
+atomic_load_acq_ ## name(volatile ptype p)
+{
+    return ((vtype)atomic_cas_acq(p, 0, 0, sz));
+}
+
+static __inline void
+atomic_store_ ## name(volatile ptype p, vtype v)
+{
+    atomic_store(p, v, sz);
+}
+static __inline void
+atomic_store_rel_ ## name(volatile ptype p, vtype v)
+{
+    atomic_store_rel(p, v, sz);
+}

-    if (__kernel_cmpxchg (prev, newval, ptr) == 0)
-        // Success.
-        return prev;
+inline jlong Atomic::load(volatile jlong* src) {
+    volatile jlong dest;
+    os::atomic_copy64(src, &dest);
+    return dest;
+}

-    // We failed even though prev == oldval. Try again.
-    }
+inline void Atomic::store(jlong store_value, jlong* dest) {
+    os::atomic_copy64((volatile jlong*)&store_value, (volatile jlong*)dest);
+}

-/* Atomically add an int to memory. */
-static inline int arm_add_and_fetch(volatile int *ptr, int add_value) {
-    for (;;) {
-        // Loop until a __kernel_cmpxchg succeeds.
+inline void Atomic::store(jlong store_value, volatile jlong* dest) {
+    os::atomic_copy64((volatile jlong*)&store_value, dest);
+}

-    int prev = *ptr;
+ATOMIC_GEN(int, u_int *, u_int, u_int, 32);
+ATOMIC_GEN(32, uint32_t *, uint32_t, uint32_t, 32);

-    if (__kernel_cmpxchg (prev, prev + add_value, ptr) == 0)

```

```

-     return prev + add_value;
- }
-}
+ATOMIC_GEN(long, u_long *, u_long, u_long, 64);
+ATOMIC_GEN(64, uint64_t *, uint64_t, uint64_t, 64);

-/* Atomically write VALUE into `*PTR' and returns the previous
- contents of `*PTR'. */
-static inline int arm_lock_test_and_set(volatile int *ptr, int newval) {
- for (;;) {
-     // Loop until a __kernel_cmpxchg succeeds.
-     int prev = *ptr;
+ATOMIC_GEN(ptr, uintptr_t *, uintptr_t, uintptr_t, 64);

-     if (__kernel_cmpxchg (prev, newval, ptr) == 0)
-         return prev;
- }
+#define         atomic_fetchadd_int         atomic_add_int
+#define         atomic_fetchadd_32         atomic_add_32
+#define         atomic_fetchadd_long       atomic_add_long
+
+#undef ATOMIC_GEN
+#undef atomic_cas
+#undef atomic_cas_acq
+#undef atomic_cas_rel
+#undef atomic_op
+#undef atomic_op_acq
+#undef atomic_op_rel
+#undef atomic_load_acq
+#undef atomic_store_rel
+#undef atomic_load_clear
+#endif
+
+static __inline __attribute__((__always_inline__))
+unsigned int atomic_add_int_nv(volatile unsigned int* dest, unsigned int add_value)
+{
+ atomic_add_acq_int(dest, add_value);
+ return *dest;
+}
+
+static __inline __attribute__((__always_inline__))
+uintptr_t atomic_add_ptr_nv(volatile intptr_t* dest, intptr_t add_value)
+{
+ atomic_add_acq_ptr((volatile uintptr_t*) dest, (uintptr_t) add_value);
+ return *((volatile uintptr_t*) dest);
+}
+
+static __inline __attribute__((__always_inline__))
+unsigned int
+atomic_swap_uint(volatile unsigned int *dest, unsigned int exchange_value)
+{
+ jint prev = *dest;
+ atomic_store_rel_int(dest, exchange_value);
+ return prev;
+}
+
+static __inline __attribute__((__always_inline__))
+void *
+atomic_swap_ptr(volatile void *dest, void *exchange_value)
+{
+ void *prev = *(void **)dest;
+ atomic_store_rel_ptr((volatile uintptr_t*) dest, (uintptr_t) exchange_value);
+ return prev;
+}
+
+static __inline __attribute__((__always_inline__))
+unsigned int
+atomic_cas_uint(volatile unsigned int *dest, unsigned int compare_value,
+ unsigned int exchange_value)
+{
+ unsigned int prev = *dest;

```

```

+ atomic_cmpset_acq_int(dest, compare_value, exchange_value);
+ return prev;
+}
+
+static __inline __attribute__((__always_inline__))
+unsigned long
+atomic_cas_ulong(volatile unsigned long *dest, unsigned long compare_value,
+ unsigned long exchange_value)
+{
+ unsigned long prev = *dest;
+ atomic_cmpset_acq_long(dest, compare_value, exchange_value);
+ return prev;
+}
+
+static __inline __attribute__((__always_inline__))
+void *
+atomic_cas_ptr(volatile void *dest, void *compare_value, void *exchange_value)
+{
+ void *prev = *(void **)dest;
+ atomic_cmpset_acq_ptr((volatile uintptr_t*) dest, (uintptr_t) compare_value, (uintptr_t) exchange_value);
+ return prev;
+}
+
+#elif defined(__APPLE__)
+
+#include <libkern/OSAtomic.h>
+
+static __inline __attribute__((__always_inline__))
+unsigned int
+atomic_add_int_nv(volatile unsigned int *target, int delta) {
+ return (unsigned int) OSAtomicAdd32Barrier(delta, (volatile int32_t *) target);
+}
+
+static __inline __attribute__((__always_inline__))
+void *
+atomic_add_ptr_nv(volatile void *target, ssize_t delta) {
+#ifdef __LP64__
+ return (void *) OSAtomicAdd64Barrier(delta, (volatile int64_t *) target);
+#else
+ return (void *) OSAtomicAdd32Barrier(delta, (volatile int32_t *) target);
+#endif
+}
+
+static __inline __attribute__((__always_inline__))
+unsigned int
+atomic_swap_uint(volatile unsigned int *dest, unsigned int exchange_value)
+{
+ /* No xchg support in OSAtomic */
+ unsigned int prev;
+ do {
+ prev = *dest;
+ } while (!OSAtomicCompareAndSwapIntBarrier((int) prev, (int) exchange_value, (volatile int *) dest));
+ return prev;
+}
+
+static __inline __attribute__((__always_inline__))
+void *
+atomic_swap_ptr(volatile void *dest, void *exchange_value)
+{
+ /* No xchg support in OSAtomic */
+ void *prev;
+ do {
+ prev = *((void * volatile *) dest);
+ } while (!OSAtomicCompareAndSwapPtrBarrier(prev, exchange_value, (void * volatile *) dest));
+ return prev;
+}
+
+static __inline __attribute__((__always_inline__))

```

```

+unsigned int
+atomic_cas_uint(volatile unsigned int *dest, unsigned int compare_value,
+ unsigned int exchange_value)
+{
+ unsigned int prev = *dest;
+ OSAtomicCompareAndSwapIntBarrier(compare_value, exchange_value, (volatile int *) dest);
+ return prev;
+}
+
+static __inline __attribute__((__always_inline__))
+unsigned long
+atomic_cas_ulong(volatile unsigned long *dest, unsigned long compare_value,
+ unsigned long exchange_value)
+{
+ unsigned long prev = *dest;
+ OSAtomicCompareAndSwapLongBarrier(compare_value, exchange_value, (volatile long *) dest);
+ return prev;
+}
+
+static __inline __attribute__((__always_inline__))
+void *
+atomic_cas_ptr(volatile void *dest, void *compare_value, void *exchange_value)
+{
+ void *prev = *(void **)dest;
+ OSAtomicCompareAndSwapPtrBarrier(compare_value, exchange_value, (void * volatile *) dest);
+ return prev;
+}
-#endif // ARM
+
+
+#endif

inline void Atomic::store(jint store_value, volatile jint* dest) {
    *dest = store_value;
@@ -171,27 +490,11 @@
}

inline jint Atomic::add(jint add_value, volatile jint* dest) {
-#ifdef ARM
- return arm_add_and_fetch(dest, add_value);
-#else
-#ifdef M68K
- return m68k_add_and_fetch(dest, add_value);
-#else
- return __sync_add_and_fetch(dest, add_value);
-#endif // M68K
-#endif // ARM
+ return (jint)atomic_add_int_nv((volatile unsigned int*) dest, add_value);
}

inline intptr_t Atomic::add_ptr(intptr_t add_value, volatile intptr_t* dest) {
-#ifdef ARM
- return arm_add_and_fetch(dest, add_value);
-#else
-#ifdef M68K
- return m68k_add_and_fetch(dest, add_value);
-#else
- return __sync_add_and_fetch(dest, add_value);
-#endif // M68K
-#endif // ARM
+ return (intptr_t)atomic_add_ptr_nv(dest, add_value);
}

inline void* Atomic::add_ptr(intptr_t add_value, volatile void* dest) {
@@ -223,95 +526,43 @@
}

inline jint Atomic::xchg(jint exchange_value, volatile jint* dest) {
-#ifdef ARM
- return arm_lock_test_and_set(dest, exchange_value);
-#else

```

```

-#ifdef M68K
- return m68k_lock_test_and_set(dest, exchange_value);
-#else
- // __sync_lock_test_and_set is a bizarrely named atomic exchange
- // operation. Note that some platforms only support this with the
- // limitation that the only valid value to store is the immediate
- // constant 1. There is a test for this in JNI_CreateJavaVM().
- return __sync_lock_test_and_set (dest, exchange_value);
-#endif // M68K
-#endif // ARM
+ return (jint)atomic_swap_uint((volatile u_int *)dest, (u_int)exchange_value);
}

inline intptr_t Atomic::xchg_ptr(intptr_t exchange_value,
                                volatile intptr_t* dest) {
-#ifdef ARM
- return arm_lock_test_and_set(dest, exchange_value);
-#else
-#ifdef M68K
- return m68k_lock_test_and_set(dest, exchange_value);
-#else
- return __sync_lock_test_and_set (dest, exchange_value);
-#endif // M68K
-#endif // ARM
+ return (intptr_t)atomic_swap_ptr((volatile void *)dest,
+ (void *)exchange_value);
}

inline void* Atomic::xchg_ptr(void* exchange_value, volatile void* dest) {
- return (void *) xchg_ptr((intptr_t) exchange_value,
- (volatile intptr_t*) dest);
+ return atomic_swap_ptr(dest, exchange_value);
}

inline jint Atomic::cmpxchg(jint exchange_value,
                            volatile jint* dest,
                            jint compare_value) {
-#ifdef ARM
- return arm_compare_and_swap(dest, compare_value, exchange_value);
-#else
-#ifdef M68K
- return m68k_compare_and_swap(dest, compare_value, exchange_value);
-#else
- return __sync_val_compare_and_swap(dest, compare_value, exchange_value);
-#endif // M68K
-#endif // ARM
+ return atomic_cas_uint((volatile u_int *)dest, compare_value, exchange_value);
}

inline jlong Atomic::cmpxchg(jlong exchange_value,
                             volatile jlong* dest,
                             jlong compare_value) {
-
- return __sync_val_compare_and_swap(dest, compare_value, exchange_value);
+ return atomic_cas_ulong((volatile u_long *)dest, compare_value,
+ exchange_value);
}

inline intptr_t Atomic::cmpxchg_ptr(intptr_t exchange_value,
                                    volatile intptr_t* dest,
                                    intptr_t compare_value) {
-#ifdef ARM
- return arm_compare_and_swap(dest, compare_value, exchange_value);
-#else
-#ifdef M68K
- return m68k_compare_and_swap(dest, compare_value, exchange_value);
-#else
- return __sync_val_compare_and_swap(dest, compare_value, exchange_value);
-#endif // M68K
-#endif // ARM
+ return (intptr_t)atomic_cas_ptr((volatile void *)dest, (void *)compare_value,

```



```

+     (void *)exchange_value);
+ }

inline void* Atomic::cmpxchg_ptr(void* exchange_value,
                                volatile void* dest,
                                void* compare_value) {
-
- return (void *) cmpxchg_ptr((intptr_t) exchange_value,
-                             (volatile intptr_t*) dest,
-                             (intptr_t) compare_value);
-}
-
-inline jlong Atomic::load(volatile jlong* src) {
- volatile jlong dest;
- os::atomic_copy64(src, &dest);
- return dest;
-}
-
-inline void Atomic::store(jlong store_value, jlong* dest) {
- os::atomic_copy64((volatile jlong*)&store_value, (volatile jlong*)dest);
-}
-
-inline void Atomic::store(jlong store_value, volatile jlong* dest) {
- os::atomic_copy64((volatile jlong*)&store_value, dest);
+ return atomic_cas_ptr((volatile void *)dest, compare_value, exchange_value);
}

-#endif // OS_CPU_LINUX_ZERO_VM_ATOMIC_LINUX_ZERO_INLINE_HPP
+#endif // OS_CPU_BSD_ZERO_VM_ATOMIC_BSD_ZERO_INLINE_HPP
--- src/os_cpu/linux_zero/vm/bytes_linux_zero.inline.hpp      2011-07-26 20:21:21.000000000 -0600
+++ src/os_cpu/bsd_zero/vm/bytes_bsd_zero.inline.hpp          2011-07-26 20:21:21.000000000 -0600
@@ -22,24 +22,31 @@
 *
 */

-#ifndef OS_CPU_LINUX_ZERO_VM_BYTES_LINUX_ZERO_INLINE_HPP
-#define OS_CPU_LINUX_ZERO_VM_BYTES_LINUX_ZERO_INLINE_HPP
+#ifndef OS_CPU_BSD_ZERO_VM_BYTES_BSD_ZERO_INLINE_HPP
+#define OS_CPU_BSD_ZERO_VM_BYTES_BSD_ZERO_INLINE_HPP

// Efficient swapping of data bytes from Java byte
// ordering to native byte ordering and vice versa.

-#include <byteswap.h>
+#ifdef __APPLE__
+#include <libkern/OSByteOrder.h>
+#define bswap16(x) OSSwapInt16(x)
+#define bswap32(x) OSSwapInt32(x)
+#define bswap64(x) OSSwapInt64(x)
+#else
+# include <sys/endian.h>
+#endif

inline u2 Bytes::swap_u2(u2 x) {
- return bswap_16(x);
+ return bswap16(x);
}

inline u4 Bytes::swap_u4(u4 x) {
- return bswap_32(x);
+ return bswap32(x);
}

inline u8 Bytes::swap_u8(u8 x) {
- return bswap_64(x);
+ return bswap64(x);
}

-#endif // OS_CPU_LINUX_ZERO_VM_BYTES_LINUX_ZERO_INLINE_HPP
+#endif // OS_CPU_BSD_ZERO_VM_BYTES_BSD_ZERO_INLINE_HPP
--- src/os_cpu/linux_zero/vm/globals_linux_zero.hpp          2011-07-26 20:21:21.000000000 -0600

```

```

+++ src/os_cpu/bsd_zero/vm/globals_bsd_zero.hpp          2011-07-26 20:21:21.000000000 -0600
@@ -23,8 +23,8 @@
 *
 */

-#ifndef OS_CPU_LINUX_ZERO_VM_GLOBALS_LINUX_ZERO_HPP
-#define OS_CPU_LINUX_ZERO_VM_GLOBALS_LINUX_ZERO_HPP
+#ifndef OS_CPU_BSD_ZERO_VM_GLOBALS_BSD_ZERO_HPP
+#define OS_CPU_BSD_ZERO_VM_GLOBALS_BSD_ZERO_HPP

//
// Set the default values for platform dependent flags used by the
@@ -46,4 +46,4 @@
// Only used on 64 bit platforms
define_pd_global(uintx, HeapBaseMinAddress,      2*G);

-#endif // OS_CPU_LINUX_ZERO_VM_GLOBALS_LINUX_ZERO_HPP
+#endif // OS_CPU_BSD_ZERO_VM_GLOBALS_BSD_ZERO_HPP
--- src/os_cpu/linux_zero/vm/orderAccess_linux_zero.inline.hpp          2011-07-26 20:21:21.000000000 -0600
+++ src/os_cpu/bsd_zero/vm/orderAccess_bsd_zero.inline.hpp              2011-07-26 20:21:21.000000000 -0600
@@ -23,8 +23,8 @@
 *
 */

-#ifndef OS_CPU_LINUX_ZERO_VM_ORDERACCESS_LINUX_ZERO_INLINE_HPP
-#define OS_CPU_LINUX_ZERO_VM_ORDERACCESS_LINUX_ZERO_INLINE_HPP
+#ifndef OS_CPU_BSD_ZERO_VM_ORDERACCESS_BSD_ZERO_INLINE_HPP
+#define OS_CPU_BSD_ZERO_VM_ORDERACCESS_BSD_ZERO_INLINE_HPP

#include "runtime/orderAccess.hpp"
#include "vm_version_zero.hpp"
@@ -46,7 +46,12 @@

#else // ARM

+#ifdef __APPLE__
+#include <libkern/OSAtomic.h>
+#define FULL_MEM_BARRIER OSMemoryBarrier()
+#else
+#define FULL_MEM_BARRIER __sync_synchronize()
+#endif // __APPLE__

+#ifdef PPC

@@ -172,4 +177,4 @@
inline void OrderAccess::release_store_ptr_fence(volatile intptr_t* p, intptr_t v) { release_store_ptr(p,
v); fence(); }
inline void OrderAccess::release_store_ptr_fence(volatile void* p, void* v) { release_store_ptr(p,
v); fence(); }

-#endif // OS_CPU_LINUX_ZERO_VM_ORDERACCESS_LINUX_ZERO_INLINE_HPP
+#endif // OS_CPU_BSD_ZERO_VM_ORDERACCESS_BSD_ZERO_INLINE_HPP
--- src/os_cpu/linux_zero/vm/os_linux_zero.cpp          2011-07-26 20:21:21.000000000 -0600
+++ src/os_cpu/bsd_zero/vm/os_bsd_zero.cpp             2011-07-26 20:21:21.000000000 -0600
@@ -23,6 +23,10 @@
 *
 */

+#if defined(_ALLBSD_SOURCE) && !defined(__APPLE__) && !defined(__NetBSD__)
+# include <pthread_np.h> /* For pthread_attr_get_np */
+#endif
+
// no precompiled headers
#include "assembler_zero.inline.hpp"
#include "classfile/classLoader.hpp"
@@ -31,11 +35,11 @@
#include "code/icBuffer.hpp"
#include "code/vtableStubs.hpp"
#include "interpreter/interpreter.hpp"
-#include "jvm_linux.h"
+#include "jvm_bsd.h"

```

```

#include "memory/allocation.inline.hpp"
-#include "mutex_linux.inline.hpp"
+#include "mutex_bsd.inline.hpp"
#include "nativeInst_zero.hpp"
-#include "os_share_linux.hpp"
+#include "os_share_bsd.hpp"
#include "prims/jniFastGetField.hpp"
#include "prims/jvm.h"
#include "prims/jvm_misc.hpp"
@@ -50,7 +54,7 @@
#include "runtime/sharedRuntime.hpp"
#include "runtime/stubRoutines.hpp"
#include "runtime/timer.hpp"
-#include "thread_linux.inline.hpp"
+#include "thread_bsd.inline.hpp"
#include "utilities/events.hpp"
#include "utilities/vmError.hpp"
#ifdef COMPILER1
@@ -102,7 +106,7 @@
    // Nothing to do.
}

-address os::Linux::ucontext_get_pc(ucontext_t* uc) {
+address os::Bsd::ucontext_get_pc(ucontext_t* uc) {
    ShouldNotCallThis();
}

@@ -117,7 +121,7 @@
}

extern "C" JNIEXPORT int
-JVM_handle_linux_signal(int sig,
+JVM_handle_bsd_signal(int sig,
                        siginfo_t* info,
                        void* ucVoid,
                        int abort_if_unrecognized) {
@@ -130,14 +134,14 @@
    // Note: it's not uncommon that JNI code uses signal/sigset to
    // install then restore certain signal handler (e.g. to temporarily
    // block SIGPIPE, or have a SIGILL handler when detecting CPU
- // type). When that happens, JVM_handle_linux_signal() might be
+ // type). When that happens, JVM_handle_bsd_signal() might be
    // invoked with junk info/ucVoid. To avoid unnecessary crash when
    // libjsig is not preloaded, try handle signals that do not require
    // siginfo/ucontext first.

    if (sig == SIGPIPE || sig == SIGXFSZ) {
        // allow chained handler to go first
-    if (os::Linux::chained_handler(sig, info, ucVoid)) {
+    if (os::Bsd::chained_handler(sig, info, ucVoid)) {
        return true;
    } else {
        if (PrintMiscellaneous && (WizardMode || Verbose)) {
@@ -151,7 +155,7 @@

    JavaThread* thread = NULL;
    VMThread* vmthread = NULL;
-    if (os::Linux::signal_handlers_are_installed) {
+    if (os::Bsd::signal_handlers_are_installed) {
        if (t != NULL) {
            if (t->is_Java_thread()) {
                thread = (JavaThread*)t;
@@ -179,15 +183,16 @@
                thread->disable_stack_red_zone();
                ShouldNotCallThis();
            }
        }
+#ifndef _ALLBSD_SOURCE
        else {
            // Accessing stack address below sp may cause SEGV if
            // current thread has MAP_GROWSDOWN stack. This should
            // only happen when current thread was created by user

```



```

-size_t os::Linux::default_guard_size(os::ThreadType thr_type) {
+size_t os::Bsd::default_guard_size(os::ThreadType thr_type) {
    // Only enable glibc guard pages for non-Java threads
    // (Java threads have HotSpot guard pages)
    return (thr_type == java_thread ? 0 : page_size());
}

static void current_stack_region(address *bottom, size_t *size) {
+ address stack_bottom;
+ address stack_top;
+ size_t stack_bytes;
+
+#ifdef __APPLE__
+ pthread_t self = pthread_self();
+ stack_top = (address) pthread_get_stackaddr_np(self);
+ stack_bytes = pthread_get_stacksize_np(self);
+ stack_bottom = stack_top - stack_bytes;
+#elif defined(__OpenBSD__)
+ stack_t ss;
+ int rslt = pthread_stackseg_np(pthread_self(), &ss);
+
+ if (rslt != 0)
+     fatal(err_msg("pthread_stackseg_np failed with err = %d", rslt));
+
+ stack_top = (address) ss.ss_sp;
+ stack_bytes = ss.ss_size;
+ stack_bottom = stack_top - stack_bytes;
+#elif defined(_ALLBSD_SOURCE)
+ pthread_attr_t attr;
+
+ int rslt = pthread_attr_init(&attr);
+
+ // JVM needs to know exact stack location, abort if it fails
+ if (rslt != 0)
+     fatal(err_msg("pthread_attr_init failed with err = %d", rslt));
+
+ rslt = pthread_attr_get_np(pthread_self(), &attr);
+
+ if (rslt != 0)
+     fatal(err_msg("pthread_attr_get_np failed with err = %d", rslt));
+
+ if (pthread_attr_getstackaddr(&attr, (void **) &stack_bottom) != 0 ||
+     pthread_attr_getstacksize(&attr, &stack_bytes) != 0) {
+     fatal("Can not locate current stack attributes!");
+ }
+
+ pthread_attr_destroy(&attr);
+
+ stack_top = stack_bottom + stack_bytes;
+else /* Linux */
+ pthread_attr_t attr;
+ int res = pthread_getattr_np(pthread_self(), &attr);
+ if (res != 0) {
@@ -326,17 +376,15 @@
+     }
+ }

- address stack_bottom;
- size_t stack_bytes;
res = pthread_attr_getstack(&attr, (void **) &stack_bottom, &stack_bytes);
if (res != 0) {
    fatal(err_msg("pthread_attr_getstack failed with errno = %d", res));
}
- address stack_top = stack_bottom + stack_bytes;
+ stack_top = stack_bottom + stack_bytes;

// The block of memory returned by pthread_attr_getstack() includes
// guard pages where present. We need to trim these off.
- size_t page_bytes = os::Linux::page_size();
+ size_t page_bytes = os::Bsd::page_size();
assert(((intptr_t) stack_bottom & (page_bytes - 1)) == 0, "unaligned stack");

```

```

size_t guard_bytes;
@@ -366,7 +414,7 @@
// The initial thread has a growable stack, and the size reported
// by pthread_attr_getstack is the maximum size it could possibly
// be given what currently mapped. This can be huge, so we cap it.
- if (os::Linux::is_initial_thread()) {
+ if (os::Bsd::is_initial_thread()) {
    stack_bytes = stack_top - stack_bottom;

    if (stack_bytes > JavaThread::stack_size_at_create())
@@ -374,6 +422,7 @@

    stack_bottom = stack_top - stack_bytes;
}
+#endif

assert(os::current_stack_pointer() >= stack_bottom, "should do");
assert(os::current_stack_pointer() < stack_top, "should do");
@@ -409,7 +458,7 @@
}

////////////////////////////////////
-// Stubs for things that would be in linux_zero.s if it existed.
+// Stubs for things that would be in bsd_zero.s if it existed.
// You probably want to disassemble these monkeys to check they're ok.

extern "C" {
--- src/os_cpu/linux_zero/vm/os_linux_zero.hpp          2011-07-26 20:21:21.000000000 -0600
+++ src/os_cpu/bsd_zero/vm/os_bsd_zero.hpp              2011-07-26 20:21:21.000000000 -0600
@@ -23,8 +23,8 @@
*
*/

-#ifndef OS_CPU_LINUX_ZERO_VM_OS_LINUX_ZERO_HPP
-#define OS_CPU_LINUX_ZERO_VM_OS_LINUX_ZERO_HPP
+#ifndef OS_CPU_BSD_ZERO_VM_OS_BSD_ZERO_HPP
+#define OS_CPU_BSD_ZERO_VM_OS_BSD_ZERO_HPP

    static void setup_fpu() {}

@@ -53,4 +53,4 @@
#endif
}

-#endif // OS_CPU_LINUX_ZERO_VM_OS_LINUX_ZERO_HPP
+#endif // OS_CPU_BSD_ZERO_VM_OS_BSD_ZERO_HPP
--- src/os_cpu/linux_zero/vm/prefetch_linux_zero.inline.hpp          2011-07-26 20:21:21.000000000 -0600
+++ src/os_cpu/bsd_zero/vm/prefetch_bsd_zero.inline.hpp              2011-07-26 20:21:21.000000000 -0600
@@ -23,8 +23,8 @@
*
*/

-#ifndef OS_CPU_LINUX_ZERO_VM_PREFETCH_LINUX_ZERO_INLINE_HPP
-#define OS_CPU_LINUX_ZERO_VM_PREFETCH_LINUX_ZERO_INLINE_HPP
+#ifndef OS_CPU_BSD_ZERO_VM_PREFETCH_BSD_ZERO_INLINE_HPP
+#define OS_CPU_BSD_ZERO_VM_PREFETCH_BSD_ZERO_INLINE_HPP

#include "runtime/prefetch.hpp"

@@ -34,4 +34,4 @@
inline void Prefetch::write(void* loc, intx interval) {
}

-#endif // OS_CPU_LINUX_ZERO_VM_PREFETCH_LINUX_ZERO_INLINE_HPP
+#endif // OS_CPU_BSD_ZERO_VM_PREFETCH_BSD_ZERO_INLINE_HPP
--- src/os_cpu/linux_zero/vm/threadLS_linux_zero.cpp          2011-07-26 20:21:21.000000000 -0600
+++ src/os_cpu/bsd_zero/vm/threadLS_bsd_zero.cpp              2011-07-26 20:21:21.000000000 -0600
@@ -25,7 +25,7 @@

#include "precompiled.hpp"

```

```

#include "runtime/threadLocalStorage.hpp"
#include "thread_linux.inline.hpp"
#include "thread_bsd.inline.hpp"

void ThreadLocalStorage::generate_code_for_get_thread() {
    // nothing to do
--- src/os_cpu/linux_zero/vm/threadLS_linux_zero.hpp          2011-07-26 20:21:21.000000000 -0600
+++ src/os_cpu/bsd_zero/vm/threadLS_bsd_zero.hpp             2011-07-26 20:21:21.000000000 -0600
@@ -22,8 +22,8 @@
*
*/

#ifndef OS_CPU_LINUX_ZERO_VM_THREADLS_LINUX_ZERO_HPP
#define OS_CPU_LINUX_ZERO_VM_THREADLS_LINUX_ZERO_HPP
#endif
#ifndef OS_CPU_BSD_ZERO_VM_THREADLS_BSD_ZERO_HPP
#define OS_CPU_BSD_ZERO_VM_THREADLS_BSD_ZERO_HPP

// Processor dependent parts of ThreadLocalStorage

@@ -32,4 +32,4 @@
    return (Thread*) os::thread_local_storage_at(thread_index());
}

#endif // OS_CPU_LINUX_ZERO_VM_THREADLS_LINUX_ZERO_HPP
#endif // OS_CPU_BSD_ZERO_VM_THREADLS_BSD_ZERO_HPP
--- src/os_cpu/linux_zero/vm/thread_linux_zero.cpp          2011-07-26 20:21:21.000000000 -0600
+++ src/os_cpu/bsd_zero/vm/thread_bsd_zero.cpp             2011-07-26 20:21:21.000000000 -0600
@@ -25,7 +25,7 @@

#include "precompiled.hpp"
#include "runtime/frame.inline.hpp"
#include "thread_linux.inline.hpp"
#include "thread_bsd.inline.hpp"

void JavaThread::cache_global_variables() {
    // nothing to do
--- src/os_cpu/linux_zero/vm/thread_linux_zero.hpp          2011-07-26 20:21:21.000000000 -0600
+++ src/os_cpu/bsd_zero/vm/thread_bsd_zero.hpp             2011-07-26 20:21:21.000000000 -0600
@@ -23,8 +23,8 @@
*
*/

#ifndef OS_CPU_LINUX_ZERO_VM_THREAD_LINUX_ZERO_HPP
#define OS_CPU_LINUX_ZERO_VM_THREAD_LINUX_ZERO_HPP
#endif
#ifndef OS_CPU_BSD_ZERO_VM_THREAD_BSD_ZERO_HPP
#define OS_CPU_BSD_ZERO_VM_THREAD_BSD_ZERO_HPP

private:
    ZeroStack _zero_stack;
@@ -118,4 +118,4 @@
    static void enable_register_stack_guard() {}
    static void disable_register_stack_guard() {}

#endif // OS_CPU_LINUX_ZERO_VM_THREAD_LINUX_ZERO_HPP
#endif // OS_CPU_BSD_ZERO_VM_THREAD_BSD_ZERO_HPP
--- src/os_cpu/linux_zero/vm/vmStructs_linux_zero.hpp      2011-07-26 20:21:21.000000000 -0600
+++ src/os_cpu/bsd_zero/vm/vmStructs_bsd_zero.hpp         2011-07-26 20:21:21.000000000 -0600
@@ -23,8 +23,8 @@
*
*/

#ifndef OS_CPU_LINUX_ZERO_VM_VMSTRUCTS_LINUX_ZERO_HPP
#define OS_CPU_LINUX_ZERO_VM_VMSTRUCTS_LINUX_ZERO_HPP
#endif
#ifndef OS_CPU_BSD_ZERO_VM_VMSTRUCTS_BSD_ZERO_HPP
#define OS_CPU_BSD_ZERO_VM_VMSTRUCTS_BSD_ZERO_HPP

// These are the OS and CPU-specific fields, types and integer
// constants required by the Serviceability Agent. This file is
@@ -47,4 +47,4 @@
/* This must be the last entry, and must be present */
last_entry()
\

```

```
-#endif // OS_CPU_LINUX_ZERO_VM_VMSTRUCTS_LINUX_ZERO_HPP
```

```
+#endif // OS_CPU_BSD_ZERO_VM_VMSTRUCTS_BSD_ZERO_HPP
```

Files `src/os_cpu/linux_zero/vm/vm_version_linux_zero.cpp` and `src/os_cpu/bsd_zero/vm/vm_version_bsd_zero.cpp` are identical