BSDPort hotspot shared changes, os

Here are the proposed JDK8 hotspot/src/os shared changes:

diff -r e6e7d76b2bd1 src/os/linux/vm/os_linux.cpp
--- a/src/os/linux/vm/os_linux.cpp        Tue May 24 15:28:35 2011 -0700
+++ b/src/os/linux/vm/os_linux.cpp        Mon Jul 25 17:04:06 2011 -0700
@@ -22,8 +22,6 @@
 /*
 */
#endif __STDC_FORMAT_MACROS
-
// no precompiled headers
#include "classfile/classLoader.hpp"
#include "classfile/systemDictionary.hpp"

diff -r e6e7d76b2bd1 src/os/posix/launcher/java_md.c
--- a/src/os/posix/launcher/java_md.c        Tue May 24 15:28:35 2011 -0700
+++ b/src/os/posix/launcher/java_md.c        Mon Jul 25 17:04:06 2011 -0700
@@ -41,14 +41,21 @@
 #include "version_comp.h"
 #endif
-#ifdef __linux__
+-#if defined(__linux__) || defined(_ALLBSD_SOURCE)
+      #include <pthread.h>
+      #else
+      #include <thread.h>
+      #endif
+      #ifdef __APPLE__
+        #define JVM_DLL "libjvm.dylib"
+        #define JAVA_DLL "libjava.dylib"
+        #define LD_LIBRARY_PATH "DYLD_LIBRARY_PATH"
+      +#else
+        #define JVM_DLL "libjvm.so"
+        #define JAVA_DLL "libjava.so"
+        #define LD_LIBRARY_PATH "LD_LIBRARY_PATH"
+      +#endif

 #ifndef GAMMA   /* launcher.make defines ARCH */
 /*
 * If not on Solaris, assume only a single LD_LIBRARY_PATH
 * variable.
 */
-  runpath = getenv("LD_LIBRARY_PATH");
+  runpath = getenv(LD_LIBRARY_PATH);
 #endif /* __sun */
-#ifdef __linux
-  #ifdef __APPLE__
+  +#elif defined(_ALLBSD_SOURCE)
+    /*
+     * On BSD, if a binary is running as sgid or suid, libc sets
+     * LD_LIBRARY_PATH to the empty string for security purposes. (In
+     * contrast, on Solaris the LD_LIBRARY_PATH variable for a
+     * privileged binary does not lose its settings; but the dynamic
+     * linker does apply more scrutiny to the path.) The launcher uses
+     * the value of LD_LIBRARY_PATH to prevent an exec loop.
+     * Therefore, if we are running sgid or suid, this function's
+     */
+     /*
+     * On BSD, if a binary is running as sgid or suid, libc sets
+     * LD_LIBRARY_PATH to the empty string for security purposes. (In
+     */
+     /*
+     * On BSD, if a binary is running as sgid or suid, libc sets
+     * LD_LIBRARY_PATH to the empty string for security purposes. (In
+     */
+     /*
+     * On BSD, if a binary is running as sgid or suid, libc sets
+     * LD_LIBRARY_PATH to the empty string for security purposes. (In
+     */
* setting of LD_LIBRARY_PATH will be ineffective and we should
* return from the function now. Getting the right libraries to
* be found must be handled through other mechanisms.
*/
if(!issetugid()) {
    return;
}
#endif
/* runpath contains current effective LD_LIBRARY_PATH setting */
new_runpath = JLI_MemAlloc( ((runpath!=NULL)?strlen(runpath):0) +
    2*strlen(jrepath) + 2*strlen(arch) +
    strlen(jvmpath) + 52);
newpath = new_runpath + strlen("LD_LIBRARY_PATH=");
newpath = new_runpath + strlen(LD_LIBRARY_PATH "=");

/* jvmpath, ((running != wanted)?((wanted==64)?"/LIBARCH64NAME":"/.."):""), */
sprintf(new_runpath, "LD_LIBRARY_PATH="
    "%s:"
    "%s/lib/%s:"
    "%s/../lib/%s",

GetApplicationHome(char *buf, jint bufsize) {
    #ifdef __linux__
    /* A utility method for asking the CPU about itself. */
    #endif
    if (execname) {
        sprintf(buf, execname, bufsize-1);
    }
    #endif /* __sun && i586 */
    #if defined(__linux__) && defined(i586)
    /* The definition of a server-class machine for bsd-i586 */
    jboolean bsd_i586_ServerClassMachine(void) {
        jboolean result = JNI_FALSE;
        /* How big is a server class machine? */
        const unsigned long server_memory = 2UL * GB;
        /* We seem not to get our full complement of memory.
        * We allow some part (1/8?) of the memory to be "missing",
        * based on the sizes of DIMMs, and maybe graphics cards.
        */
        const unsigned long missing_memory = 256UL * MB;
        const unsigned long actual_memory = physical_memory();
        /* Is this a server class machine? */
        if (actual_memory >= (server_memory - missing_memory)) {
            const unsigned long actual_processors = physical_processors();
            if (actual_processors >= server_processors) {
result = JNI_TRUE;
+
+
+
+
if (_launcher_debug) {
  printf("linux_" LIBARCHNAME "._ServerClassMachine: \s\n",
        (result == JNI_TRUE ? "true" : "false");
  }
return result;
+
+
+#endif /* _ALLBSD_SOURCE && i586 */
+
+/* Dispatch to the platform-specific definition of "server-class" */
+jboolean
ServerClassMachine(void) {
@@ -1466,6 +1522,8 @@
  result = solaris_i586_ServerClassMachine();
  #elif defined(__linux__) && defined(i586)
    result = linux_i586_ServerClassMachine();
+  #elif defined(_ALLBSD_SOURCE) && defined(i586)
+    result = bsd_i586_ServerClassMachine();
  #else
    if (_launcher_debug) {
      printf("ServerClassMachine: returns default value of \s\n",
@@ -1821,7 +1879,7 @@
int
ContinueInNewThread(int (JNICALL *continuation)(void *), jlong stack_size, void * args) {
  int rslt;
-#ifdef __linux__
+#elif defined(__linux__) || defined(_ALLBSD_SOURCE)
    pthread_t tid;
    pthread_attr_t attr;
    pthread_attr_init(&attr);