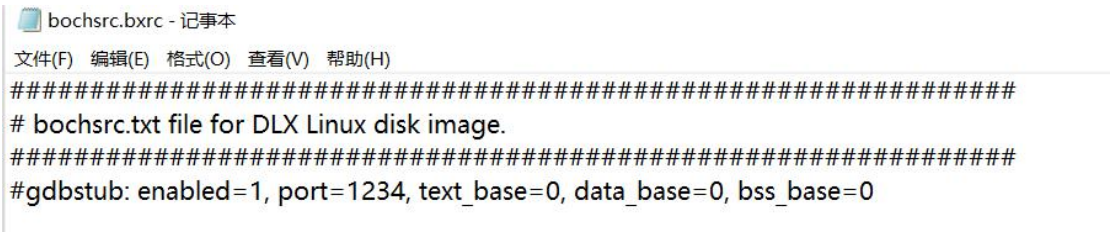
P01：UNIX V6++运行调制环境的安装与配置

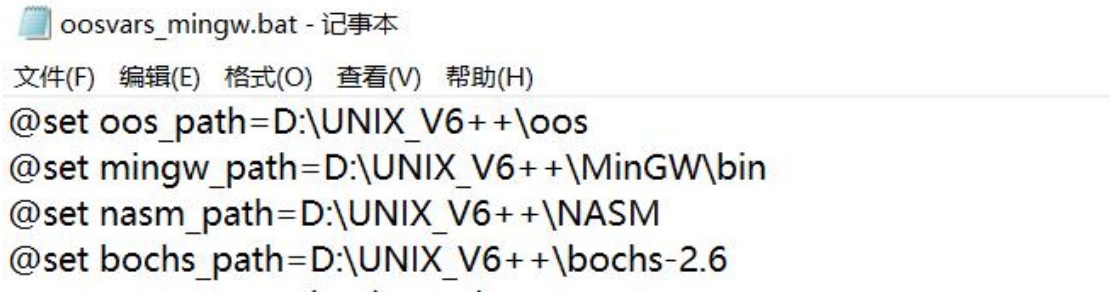
2152118 史君宝

一、 UNIX V6++的运行环境的安装与配置

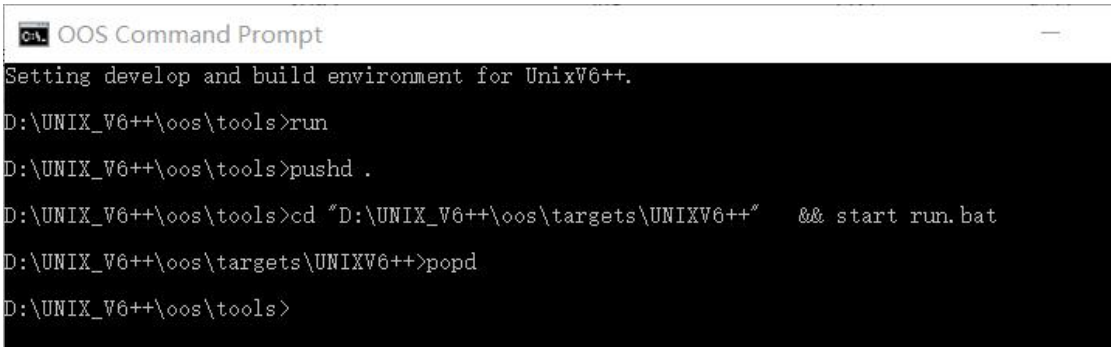
（1） 修改 bochsrc.bxrc 文件，去除 bochs 虚拟机的 gdb 调试功能。

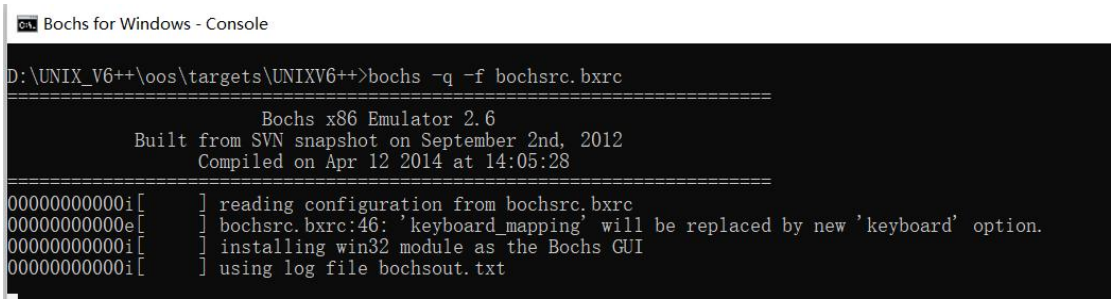


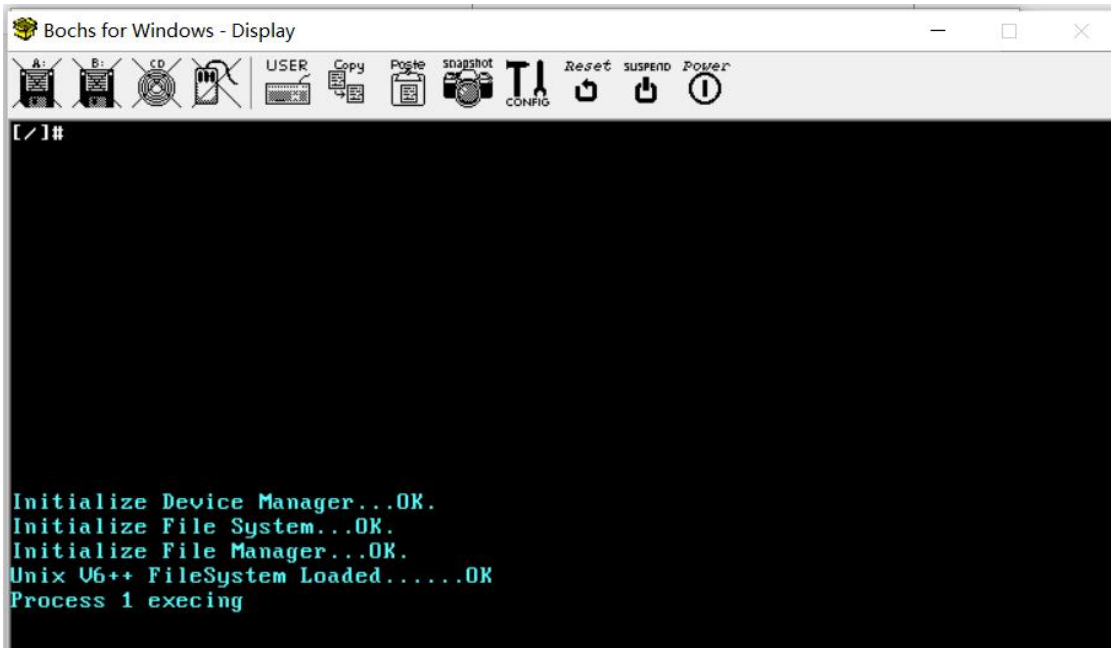
（2） 运行 UNIX V6++的实验环境，修改各工具的路径信息。



（3） 运行 UNIX V6++环境。

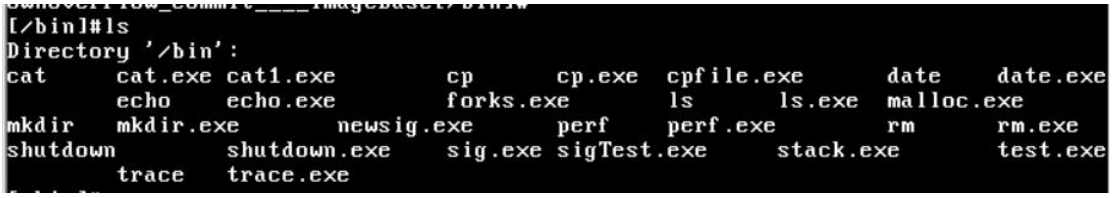






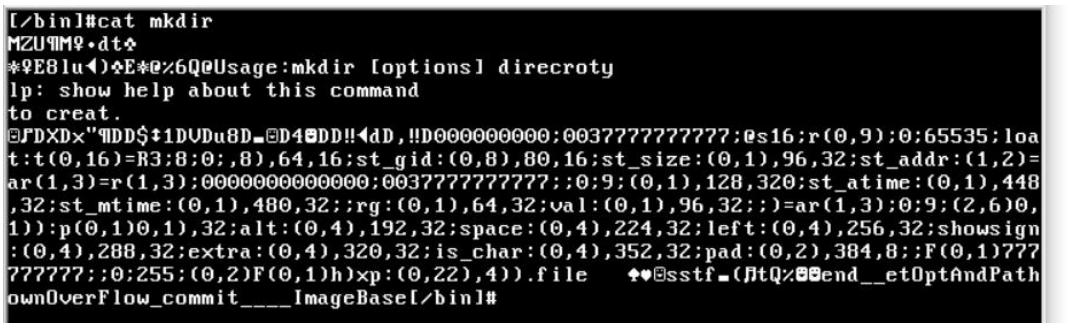
（4） Unix V6++指令的使用

Unix V6++系统中的所有指令：



Cat 指令 使用方式：

由上我们可以知道其具体的语法：
我们看到 cat 指令可以输出一个文件的具体内容，由于其他两个创建文件和合并文件的功能可能会影响其内容及结构，在这里就不再演示了。



CP 指令 使用方式

```
Usage:cp [options] filename1 filename2 ... targetfile
Options:
```

这个指令能够实现文件的复制，为保证内容的不变在这里也不再演示了。

Date 指令 使用方式

```
[/bin]#date
24-Sep-2023 11:12:49(SUN)
[/bin]#
```

这个指令能够直接显示当前的日期和时间。

echo 指令 使用方式

```
[/bin]#echo enang
enang
[/bin]#
```

这个指令能够在终端输出文本或者变量的值。

Ls 指令 使用方式：

```
[/bin]#ls
Directory '/bin':
cat      cat.exe cat1.exe        cp      cp.exe  cpfile.exe      date    date.exe
         echo    echo.exe        forks.exe       ls      ls.exe  malloc.exe
mkdir    mkdir.exe       newsig.exe      perf    perf.exe        rm      rm.exe
shutdown        shutdown.exe    sig.exe sigTest.exe     stack.exe       test.exe
         trace   trace.exe
```

该指令能够输出当前文件夹下的所有文件目录。

二、 UNIX V6++的 eclipse 远程调试环境的安装置

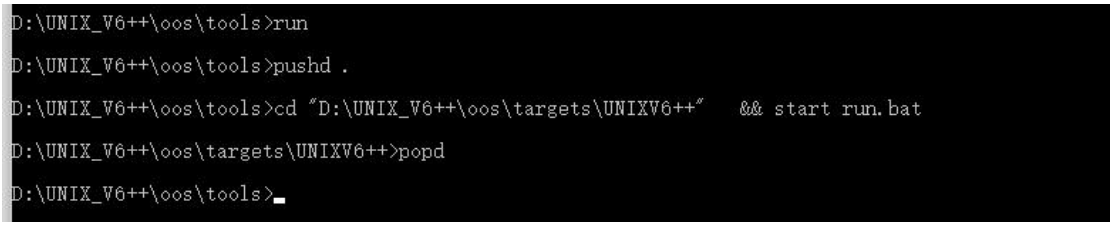
（1） 修改 bochsrc.bxrc 文件，增加 bochs 虚拟机的 gdb 调试功能。

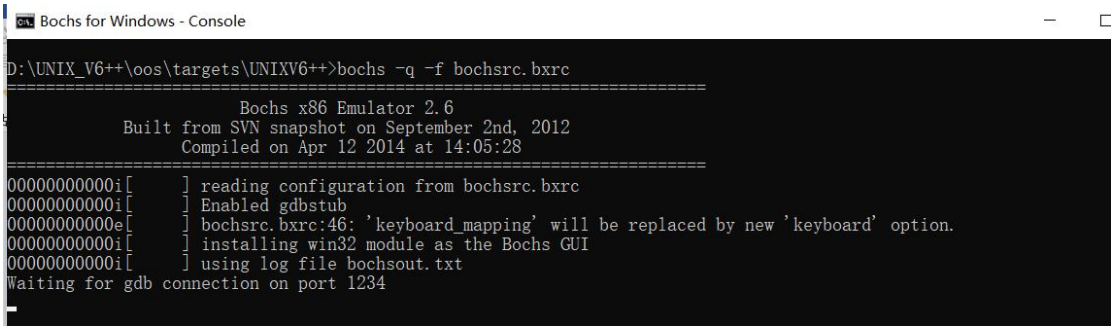
bochsrc.bxrc - 记事本

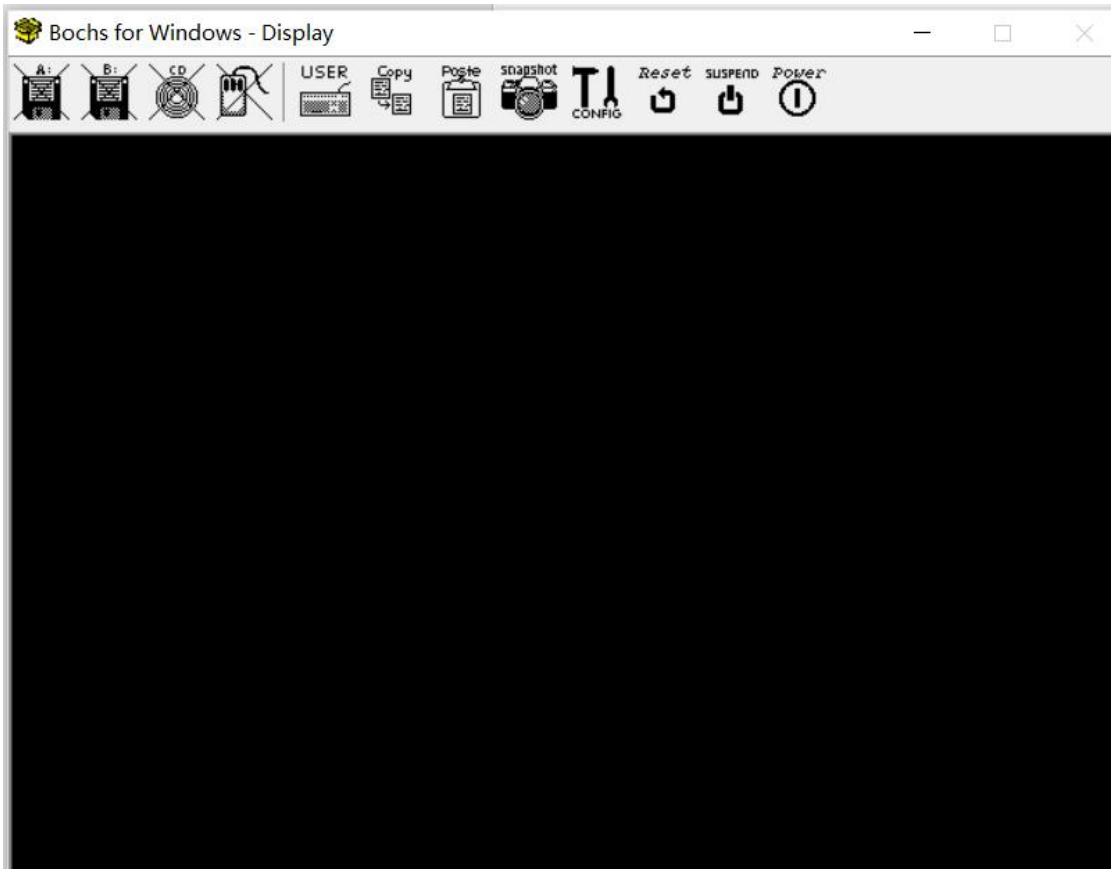
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
#######################################################################
# bochsrc.txt file for DLX Linux disk image.
#######################################################################
gdbstub: enabled=1, port=1234, text_base=0, data_base=0, bss_base=0
```

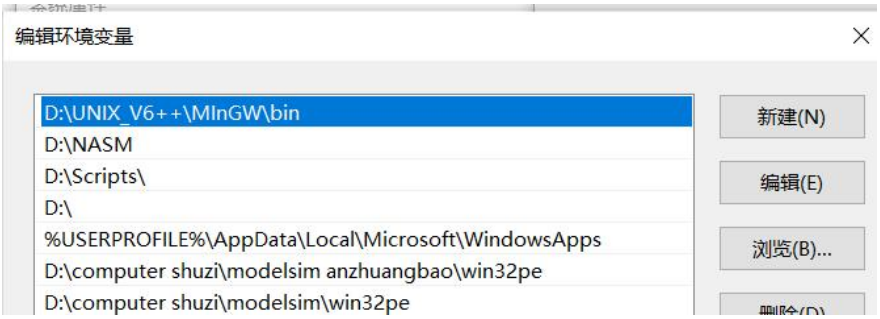
（2） 再次运行 UNIX V6++，出现等待窗口的界面。


D:\UNIX_V6++\oos\tools>run
D:\UNIX_V6++\oos\tools>pushd .
D:\UNIX_V6++\oos\tools>cd "D:\UNIX_V6++\oos\targets\UNIXV6++" && start run.bat
D:\UNIX_V6++\oos\targets\UNIXV6++>popd
D:\UNIX_V6++\oos\tools>

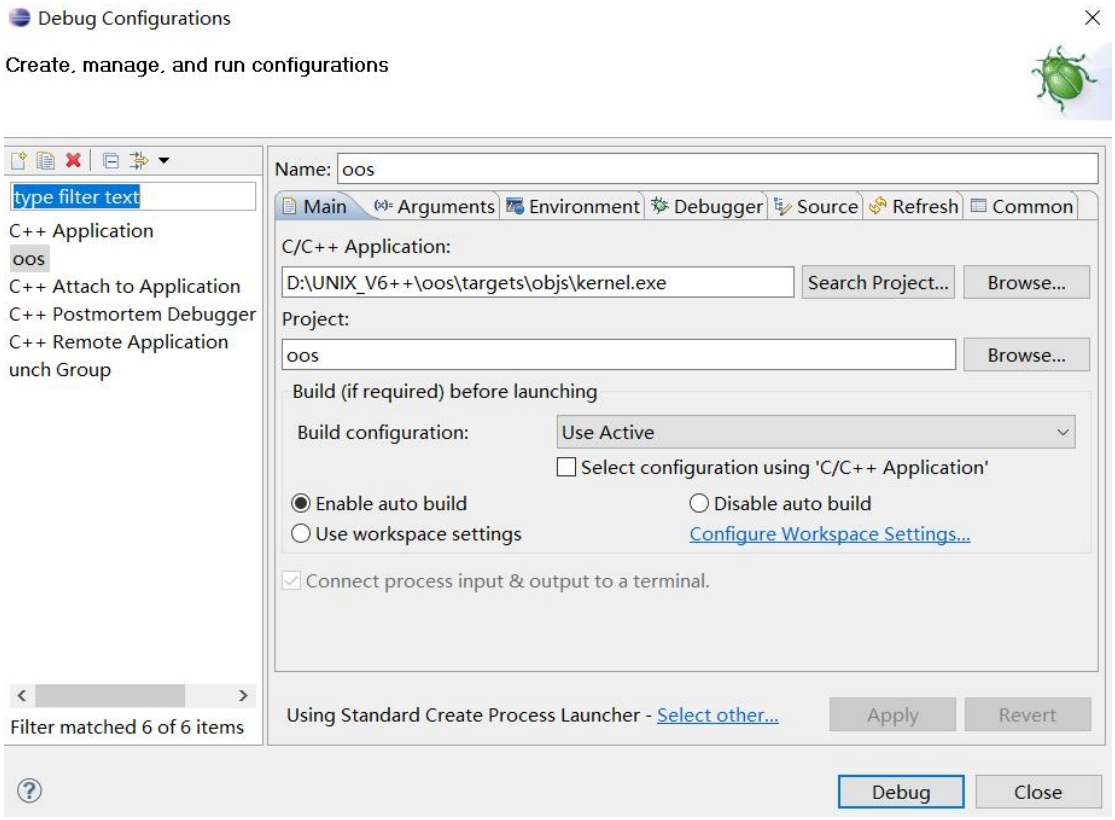

Bochs for Windows - Console
D:\UNIX_V6++\oos\targets\UNIXV6++>bochs -q -f bochsrc.bxrc
Bochs x86 Emulator 2.6
Built from SVN snapshot on September 2nd, 2012
Compiled on Apr 12 2014 at 14:05:28
00000000000i[ ] reading configuration from bochsrc.bxrc
00000000000i[ ] Enabled gdbstub
00000000000e[ ] bochsrc.bxrc:46: 'keyboard_mapping' will be replaced by new 'keyboard' option.
00000000000i[ ] installing win32 module as the Bochs GUI
00000000000i[ ] using log file bochsout.txt
Waiting for gdb connection on port 1234

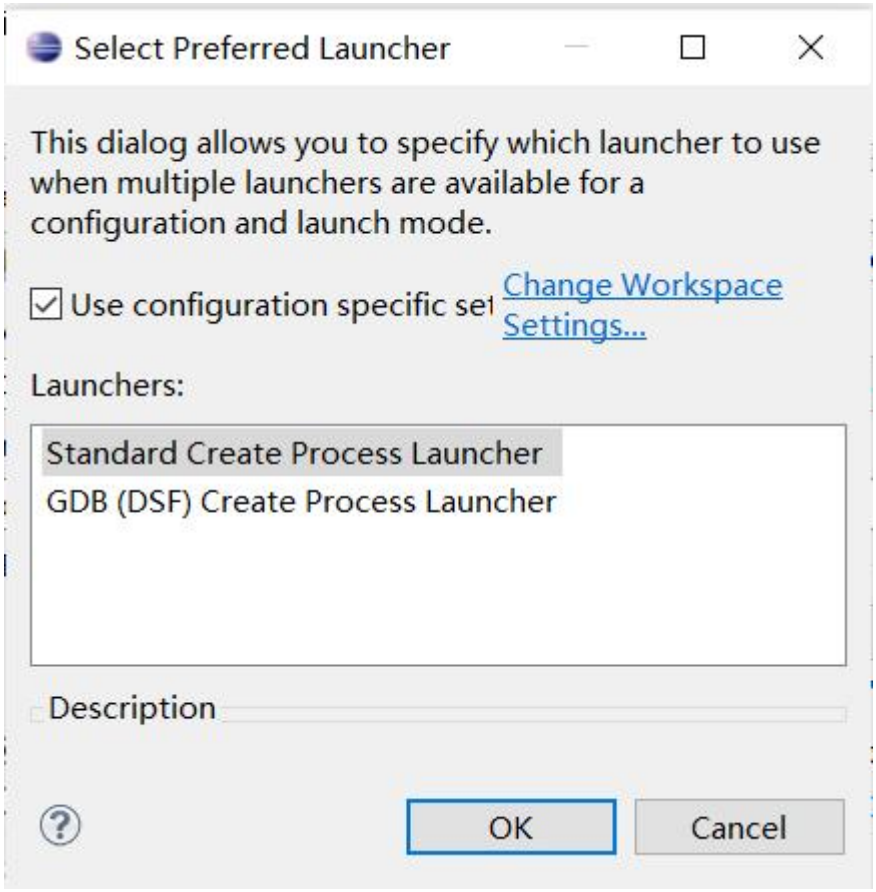

Bochs for Windows - Display
A:
B:
CD
USER
Copy
Paste
snapshot
CONFIG
Reset
SUSPEND
Power

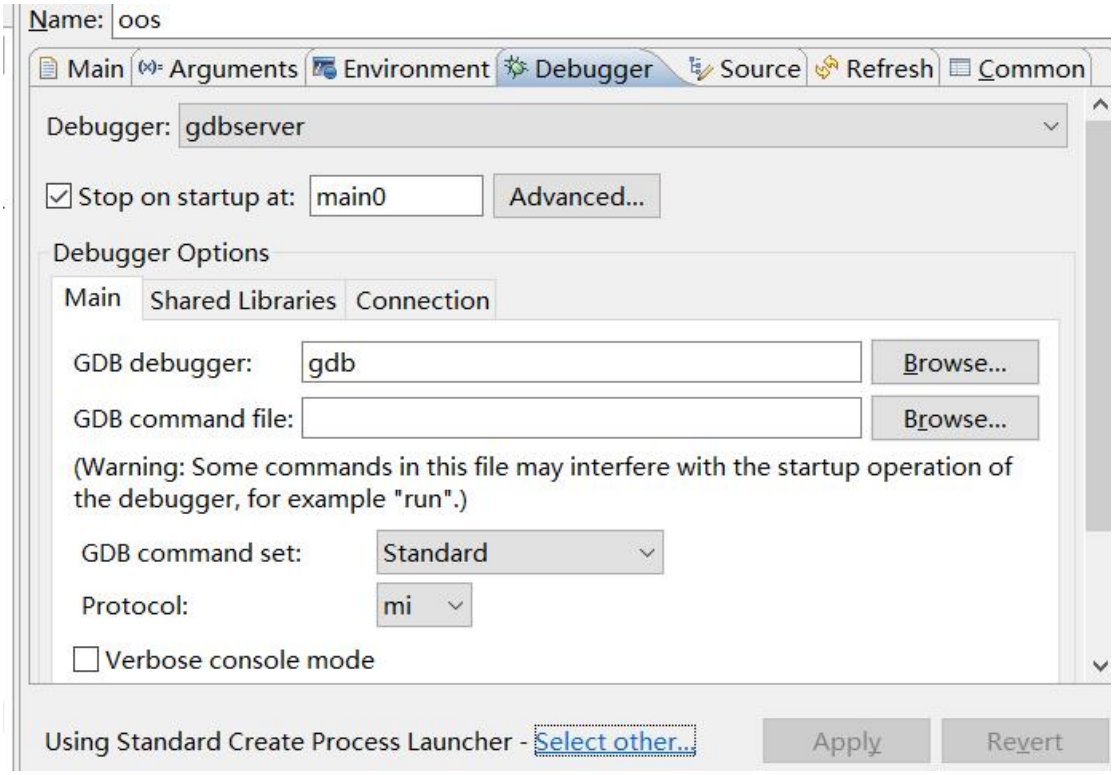
（3） 系统环境变量的配置。

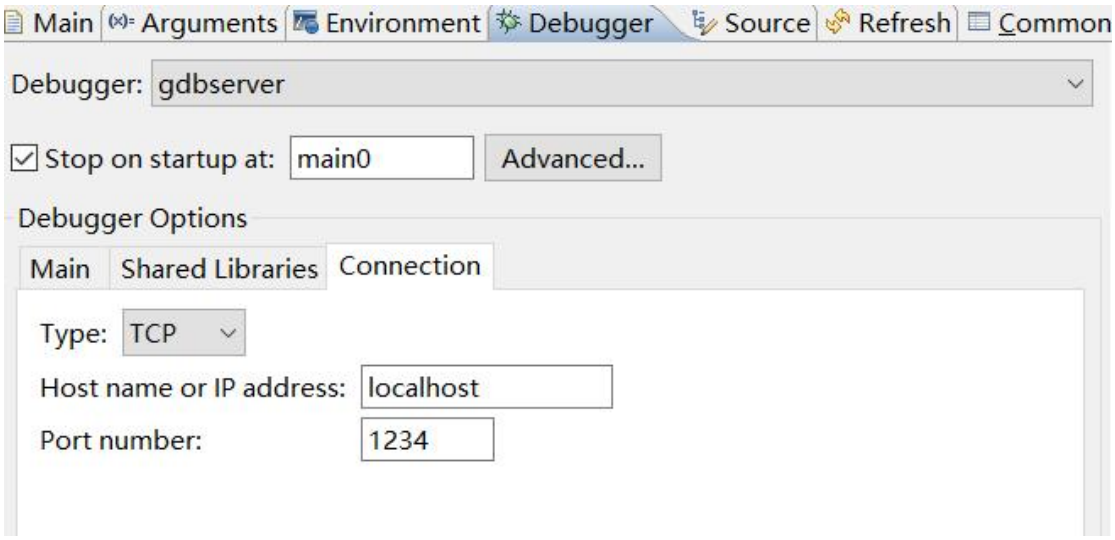

编辑环境变量
D:\UNIX_V6++\MInGW\bin
D:\NASM
D:\Scripts\
D:\
%USERPROFILE%\AppData\Local\Microsoft\WindowsApps
D:\computer shuzi\modelsim anzhuangbao\win32pe
D:\computer shuzi\modelsim\win32pe
新建(N)
编辑(E)
浏览(B)...

（4） 打开 eclipse 进行配置。



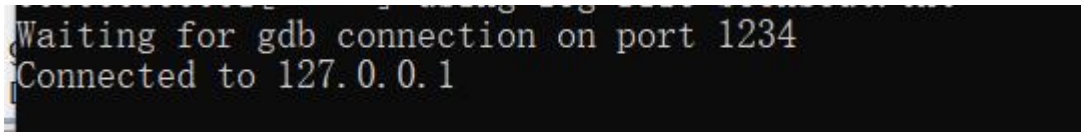


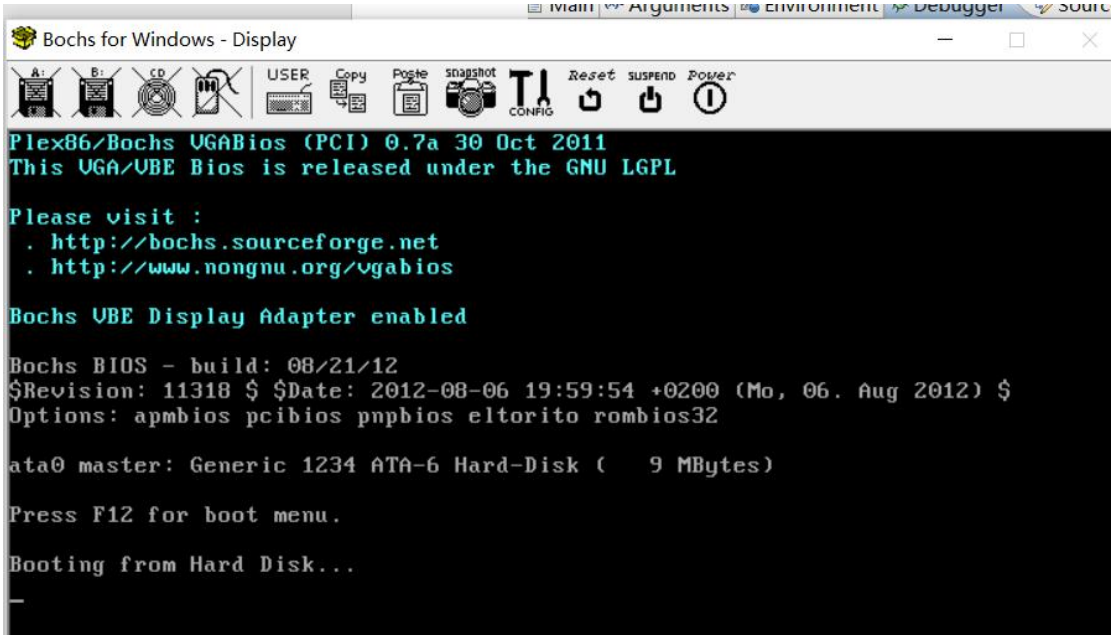




（5） 重新启动 UNIX V6++，并启动调试，观察现象

出现连接成功的现象





三、 UNIX V6++的调试运行

（1）设置断点一：

```
    //init gdt
    machine.InitGDT();
    machine.LoadGDT();
    //init idt
    machine.InitIDT();
    machine.LoadIDT();

    //init page protection
    machine.InitPageDirectory();
    machine.EnablePageProtection();
```

变量值：

Name	Value
machine	{...}
KERNEL_CODE_SEGMENT_SELECTOR	0
KERNEL_DATA_SEGMENT_SELECTOR	0
USER_CODE_SEGMENT_SELECTOR	0
USER_DATA_SEGMENT_SELECTOR	0
TASK_STATE_SEGMENT_SELECTOR	0
TASK_STATE_SEGMENT_IDX	0
PAGE_DIRECTORY_BASE_ADDRESS	
KERNEL_PAGE_TABLE_BASE_ADDRESS	
USER_PAGE_TABLE_BASE_ADDRESS	
USER_PAGE_TABLE_CNT	
KERNEL_SPACE_SIZE	0
KERNEL_SPACE_START_ADDRESS	
instance	{...}
m_IDT	<incomplete type>
m_GDT	<incomplete type>
m_PageDirectory	0x00ff53f0
m_KernelPageTable	16733168
m_UserPageTable	16733168
m_TaskStateSegment	16733168

寄存器值：

Variables | Breakpoints | Registers | Modules

Name	Value
Main	
eax	39
ecx	-1072571200
edx	-1072571200
ebx	1140736
esp	0xc000ffca
ebp	0xc000ffd2
esi	917504
edi	65452
eip	0xc010904b
eflags	[PF SF]
cs	24
ss	32
ds	32
es	32
fs	0
gs	0
st0	0
st1	0
st2	0

（2）设置断点二：

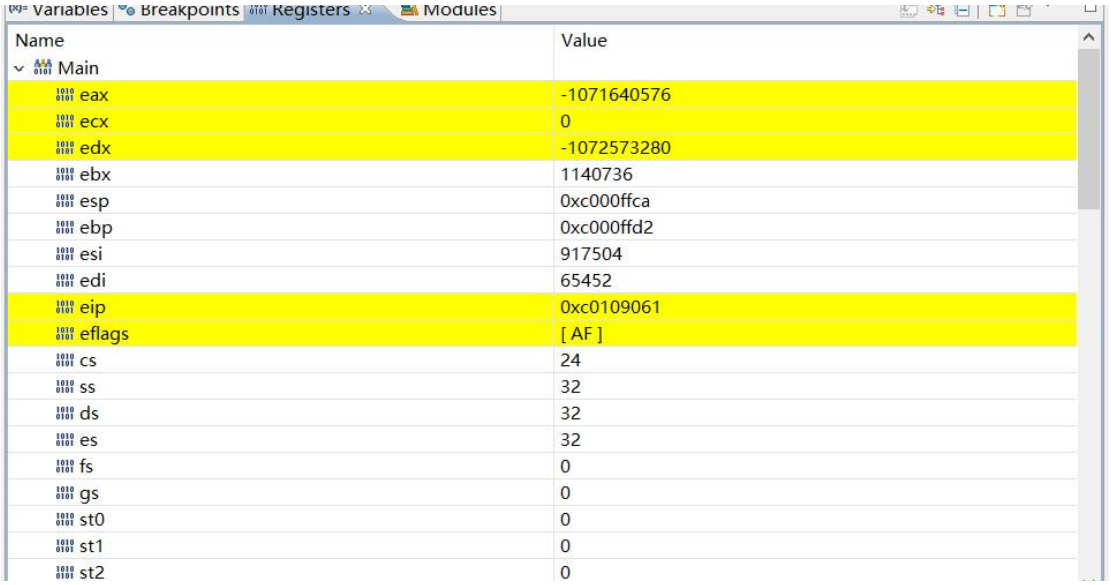
```
    machine.LoadIDT();

    //init page protection
    machine.InitPageDirectory();
    machine.EnablePageProtection();
    /*
     * InitPageDirectory()中将线性地址0-4M映射到物理内存
     * 0-4M是为保证此注释以下至本函数结尾的代码正确执行!
     */

    //使用0x10段寄存器
    __asm__ __volatile__
```

变量值：

Variables | Breakpoints | Registers | Modules

Name	Value
machine	{...}
KERNEL_CODE_SEGMENT_SELECTOR	0
KERNEL_DATA_SEGMENT_SELECTOR	0
USER_CODE_SEGMENT_SELECTOR	0
USER_DATA_SEGMENT_SELECTOR	0
TASK_STATE_SEGMENT_SELECTOR	0
TASK_STATE_SEGMENT_IDX	0
PAGE_DIRECTORY_BASE_ADDRESS	
KERNEL_PAGE_TABLE_BASE_ADDRESS	
USER_PAGE_TABLE_BASE_ADDRESS	
USER_PAGE_TABLE_CNT	
KERNEL_SPACE_SIZE	0
KERNEL_SPACE_START_ADDRESS	
instance	{...}
m_IDT	<incomplete type>
m_GDT	<incomplete type>
m_PageDirectory	0x00ff53f0
m_KernelPageTable	16733168
m_UserPageTable	16733168
m_TaskStateSegment	16733168

寄存器值：



四、 UNIX V6++源代码目录结构

boot/：应该是包含引导加载程序的源代码和配置文件。

dev/：应该是包含设备驱动程序的源代码，用于与硬件设备进行交互。

fs/：应该是包含文件系统相关的源代码，包括文件系统的实现和文件操作。

include/：包含系统的头文件，就像我们在 C++ 中使用的 include <iostream> 一样定义了各种常量、数据结构和函数原型等。

kernel/：包含内核的核心代码，整个操作系统最重要的部分，负责处理系统调用、进程管理、内存管理等核心功能。

lib/：包含通用的库函数和工具函数的源代码，就像我们在使用 eclipse 中导入的库一样。

mm/：应该与存储相关，包含内存管理相关的源代码，比如内存分配、虚拟内存管理等等。

sys/：应该包含有关系统调用的相关信息，比如包含系统调用相关的源代码，定义了系统调用的接口和实现。

tools/：包含一些辅助工具和实用程序的源代码，用于构建和调试系统。