实验二：在 UNIX V6++中添加新的系统调用

# 实验目的

* 通过阅读汇编程序，观察 C++程序运行时栈、寄存器和内存变量的变化，理解 C/C++编译器对函数调用的处理过程。
* 结合课程所学知识，通过在 UNIX V6++源代码中添加一个新的系统调用，熟悉UNIX V6++中系统调用相关部分的程序结构。
* 通过实践，进一步掌握 UNIX V6++重新编译及运行调试的方法。

# 实验要求

* 完成实验 4.1，截图说明编译形成的维护栈帧的汇编指令的作用。
* 完成实验 4.2~4.5，按上述过程向 UNIX V6++中添加一个新的系统调用， 要求实验报告中说明新的系统调用的功能和运行的效果（请选择需附代码和截图）。
* 完成实验 4.6，在调试环境下单步执行，观察 UNIX V6++的调试运行， 截图说明变量与寄存器的值的变化，特别是 EAX 寄存器的值。

# 实验内容

## 程序运行时环境：栈、寄存器和内存变量

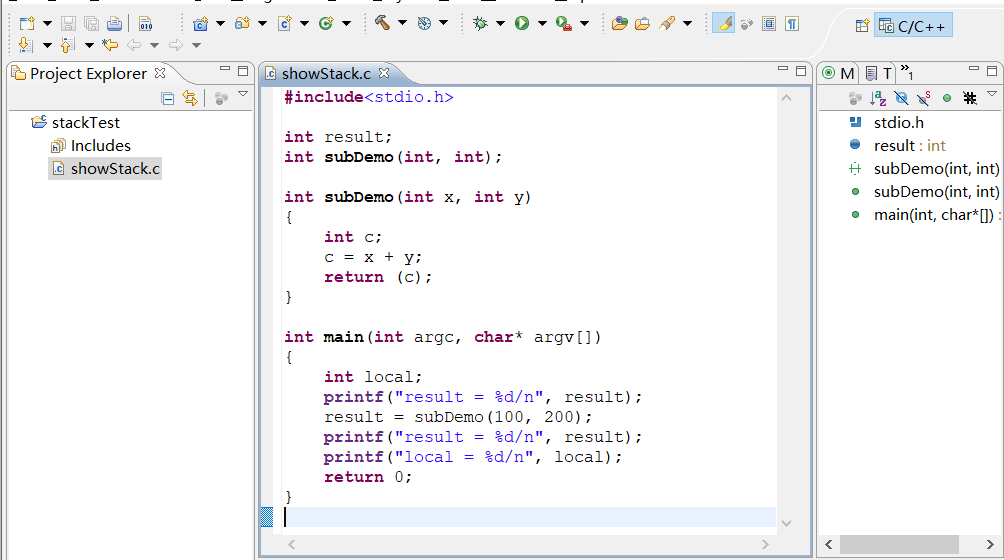
编写一个简单的 C 语言程序调用，观察汇编代码对堆栈的修改。具体过程如下。

### 新建一个 C 工程

在 eclipse 中新建一个工程，在该工程中添加一个.c 源文件， 并在该源文件中如图 5 所示的源。

|  |  |
| --- | --- |
| 新建工程 | 新建源文件 |

图3-1

图3-2 写入源代码

### 编译与调试运行

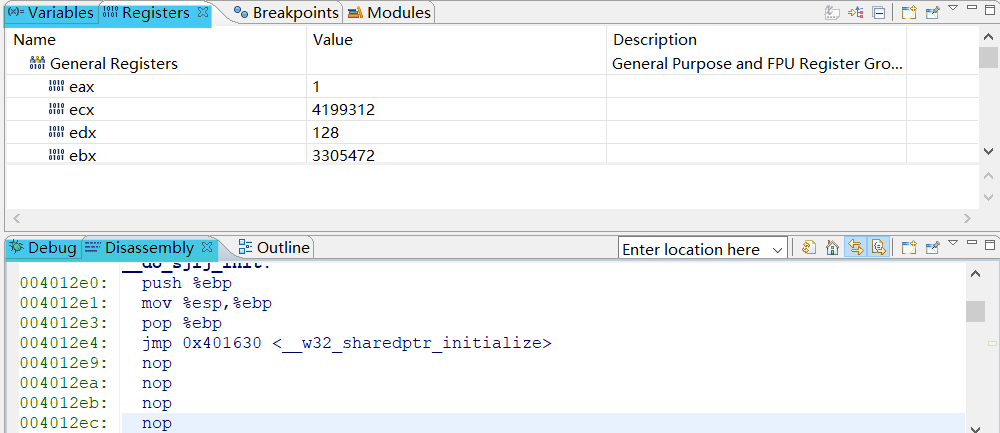
编译该工程。打开 Debug 视图、反汇编视图、Variable 视图和 Register 视图。表里没有的，可以到 Others 里面去找。

图3-3 打开视图

下断点（比如 15 行）之后开始调试。可以看到 subDemo 函数的汇编指令和所有寄存器的当前值。按 Step into，观察程序运行时，EIP、ESP、EBP 和 EAX 的变化。

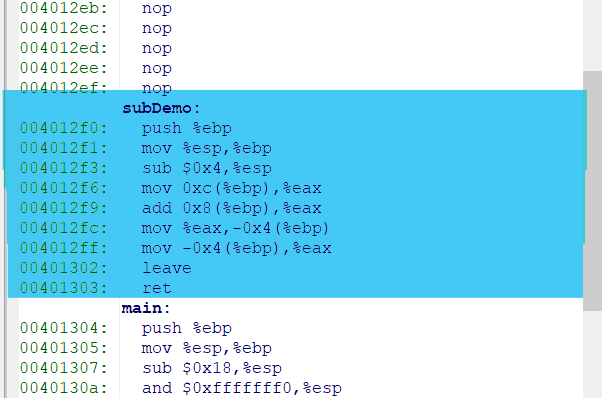


图3-4 汇编指令

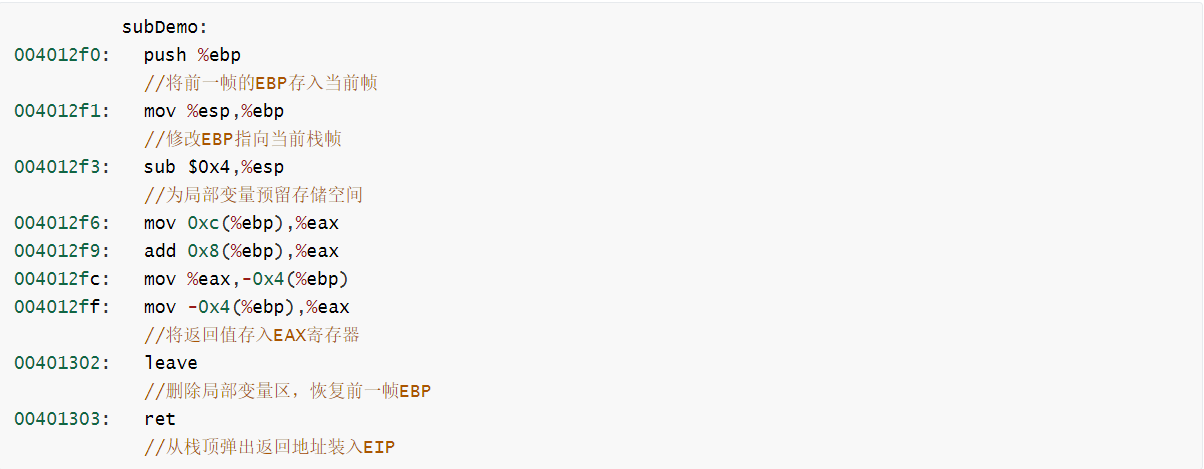
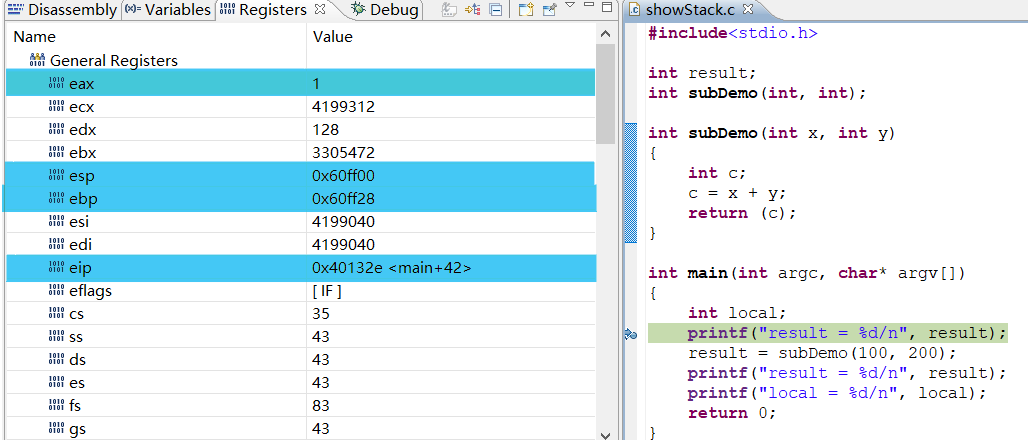
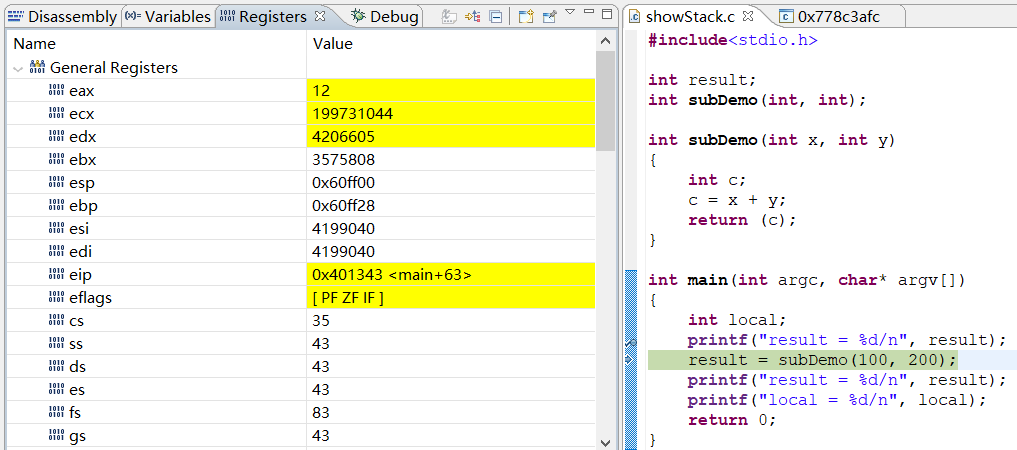
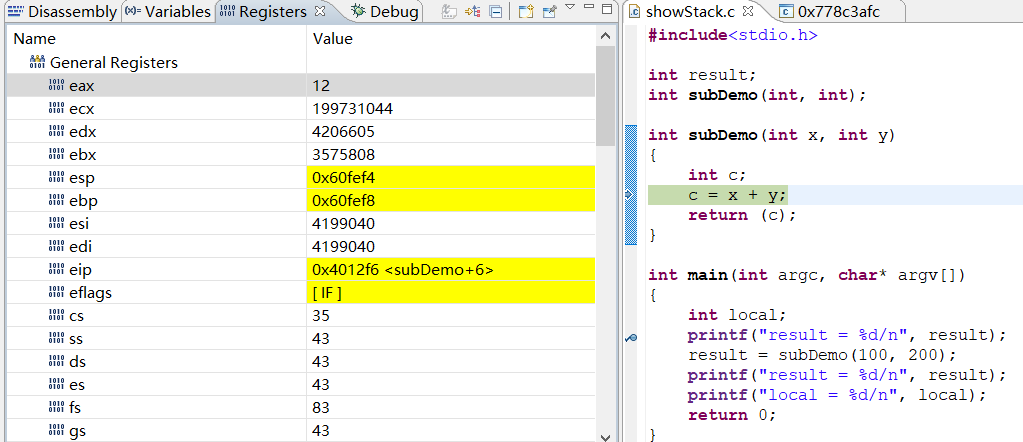
说明编译形成的维护栈帧的汇编指令的作用：

图3-5 维护栈帧的汇编指令作用

观察程序运行时，EIP、ESP、EBP 和 EAX 的变化。







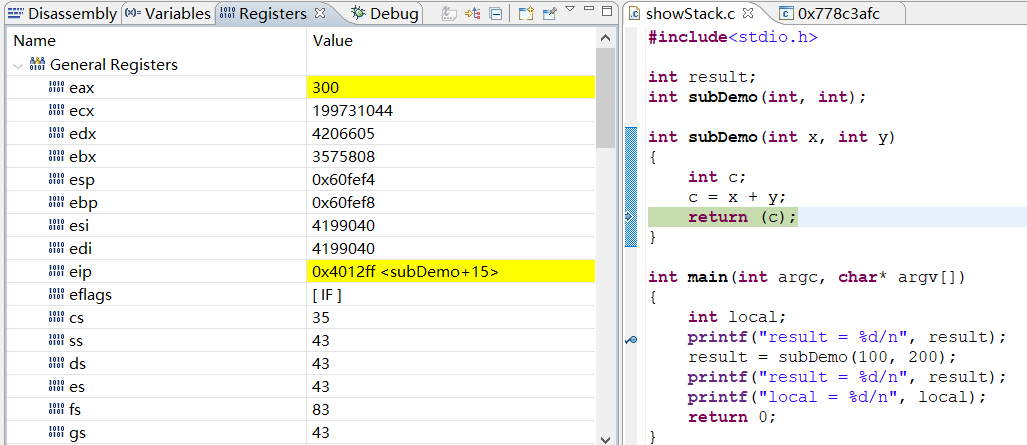


图3-6 4个寄存器的变化

## 在 UNIX V6++中添加一个新的系统调用接口

### 在 SystemCall 类中添加系统调用处理子程序的定义

首先在SystemCall.h文件中添加一个新的系统调用处理子程序的声明

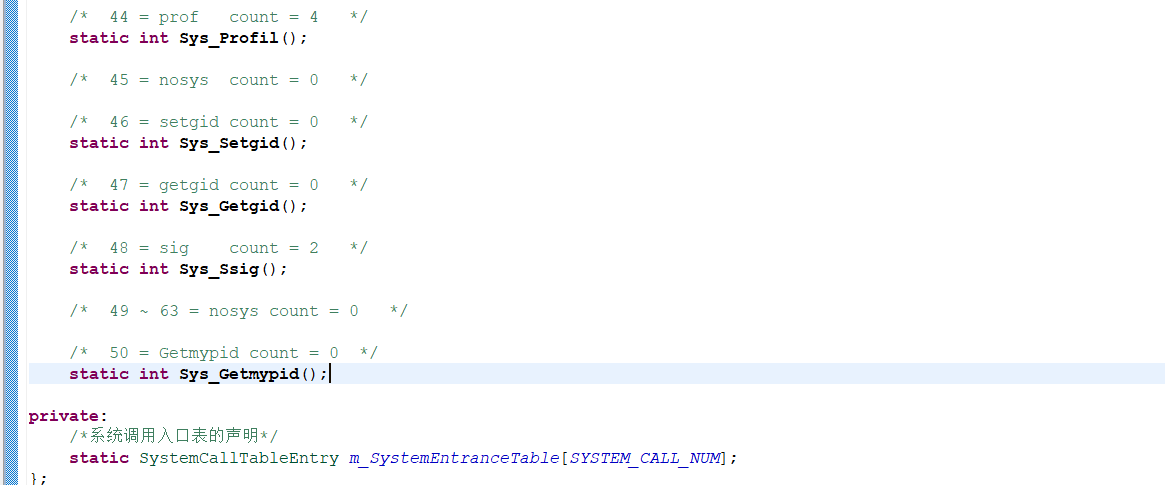


图3-7：在 SystemCall.h 中添加新的系统调用处理子程序的声明

其次，在 SystemCall.cpp 中添加 Sys\_Getmypid 的定义

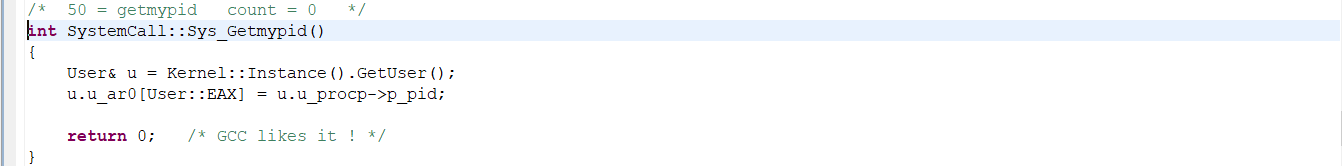


图 3-8：在 SystemCall.h 中添加新的系统调用处理子程序的声明

### 在新的系统调用处理子程序加入系统调用子程序入口表中

在 SystemCall.cpp 中找到对系统调用子程序入口表 m\_SystemEntranceTable 赋值的一段程序代码，如下：

SystemCallTableEntry SystemCall::m\_SystemEntranceTable[SYSTEM\_CALL\_NUM] = …… 选择第 50 项，并用{ 0, &Sys\_Getmypid}来替换原来的{ 0, &Sys\_Nosys }，即：第 50 号系统调用所需参数为 0 个，系统调用处理子程序的入口地址为：&Sys\_Getmypid。

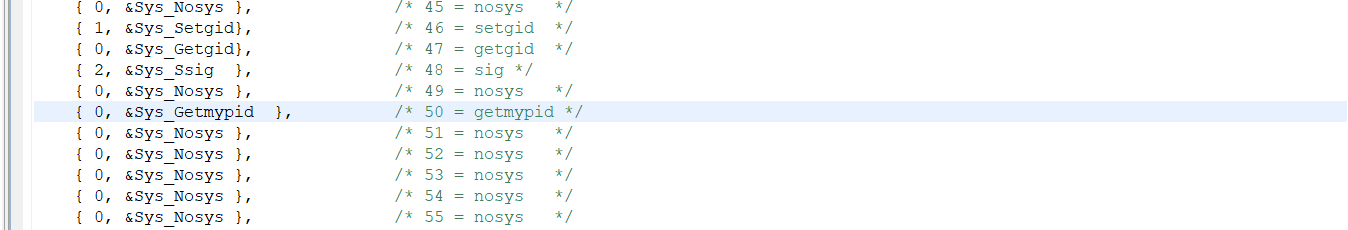
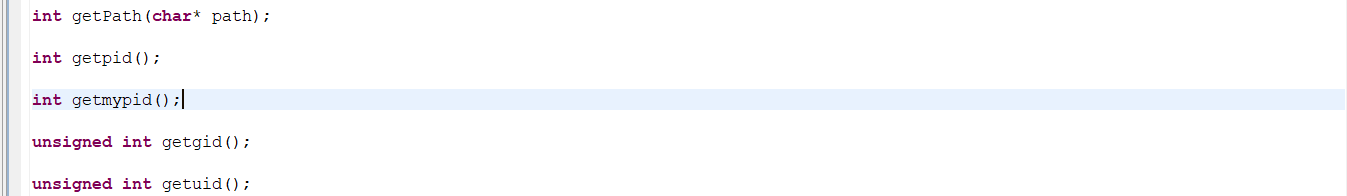


图3-9 将新定义的系统调用处理程序加入处理程序入口表中

## 为新的系统调用添加对应的库函数

### 在 sys.h 文件中添加库函数的声明

找到 sys.h 文件，在其中加入名为 getmypid 的库函数的声明（如图 3-10所示）。

图3-10 添加库函数的声明

### 在 sys.c 中添加库函数的定义

在 sys.c 文件中添加库函数 getmypid 的定义需要根据自己定义的系统调用在子程序入口表中的实际位置，填入正确的系统调用号。

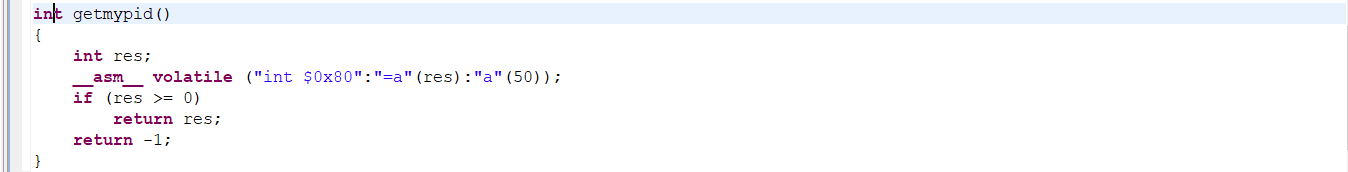


图3-11 添加库函数的定义

## 编写测试程序

在 UNIX V6++中添加一个可执行程序，需要在 src/program 文件夹下添加一个源程序文件，并编译通过之后，才可以运行。

### 在 program 文件加入一个新的 c 语言文件

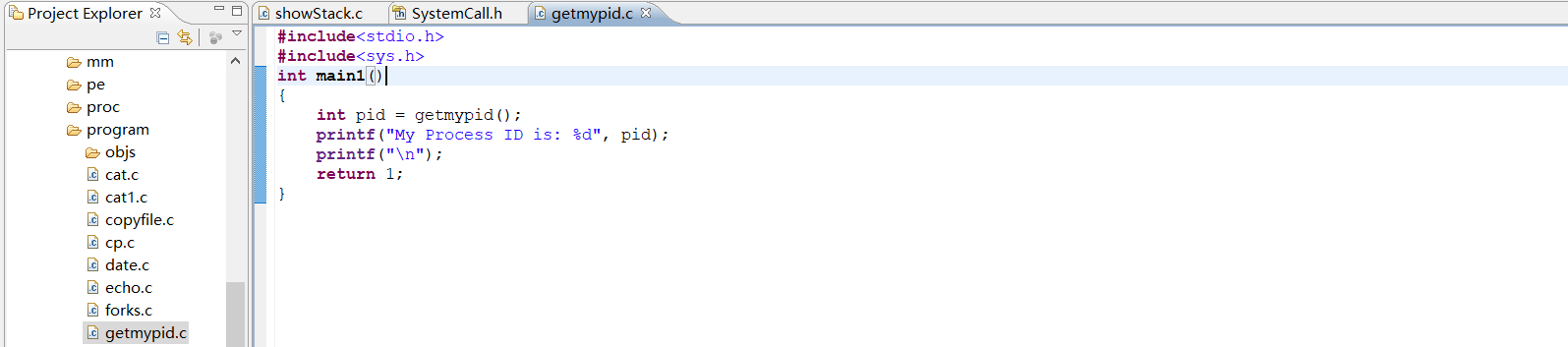
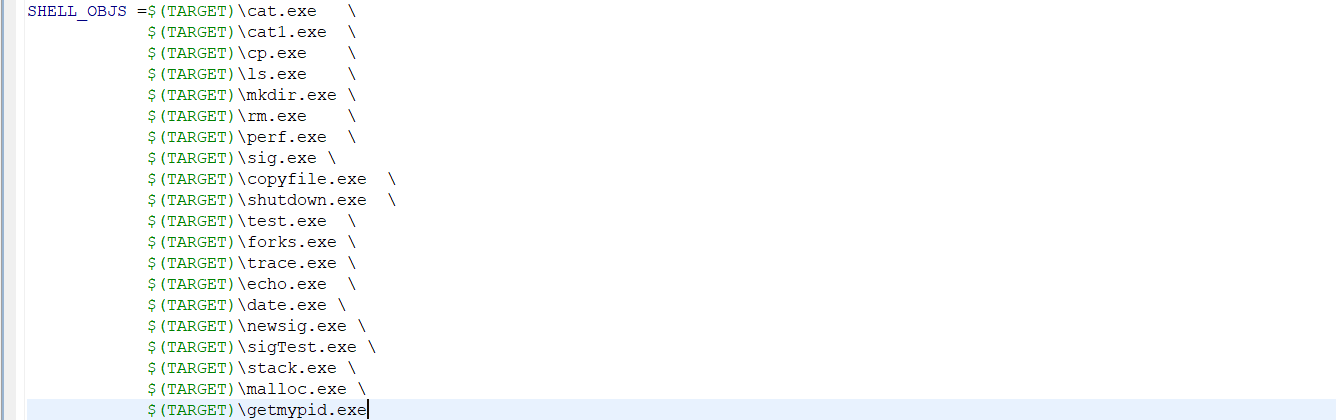


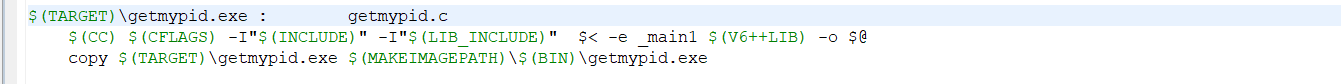
图3-12 加入新的C语言文件

### 修改编译需要使用的 makefile 文件

为了完成上述测试程序源文件的编译，需要修改 program 文件夹下的 Makefile 文件。



(a)



(b)

图3-13 修改MAKEFILE文件

## 重新编译 UNIX V6++代码

eclipse 中选择 project-Build All，完成对 UNIX V6++代码的重新编译。如果编译成功，则启动 UNIX V6++之后，进入 bin 文件夹，可以看到该文件夹下有刚编译通过形成的可执行文件 getmypid。键入 getmypid 可以看到屏幕输出当前的进程号。

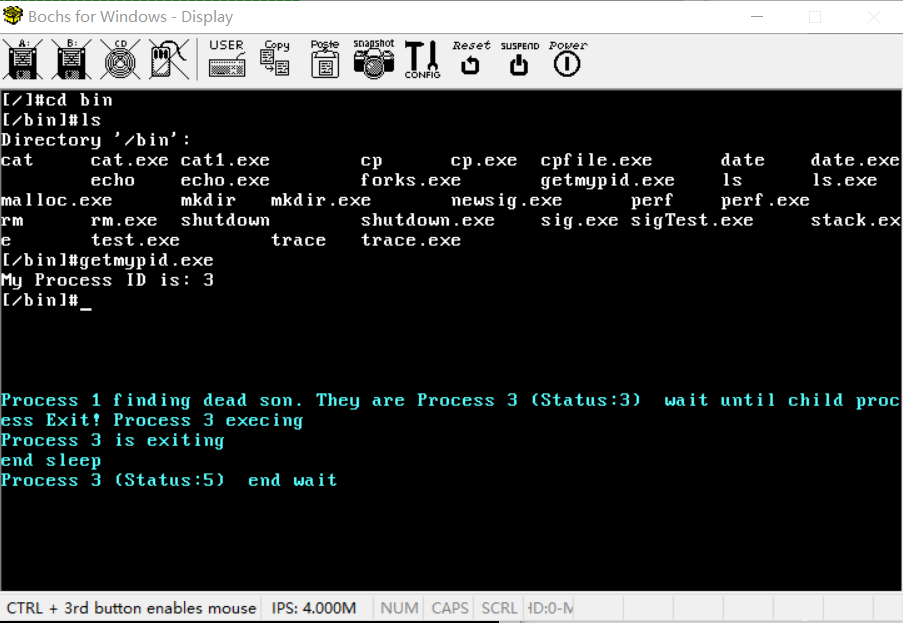


图 3-14：测试程序运行结果

### 新的系统调用的功能和运行的效果

新的系统调用的功能为返回当前运行进程的进程号（通过 EAX 寄存器带回当前进程的 p\_pid），运行效果如图3-14所示输出进程号 3。

## 程序调试

可以在程序运行过程中添加断点，通过调试观察程序的运行情况。例如：可以在如图3-15 所示位置添加断点。

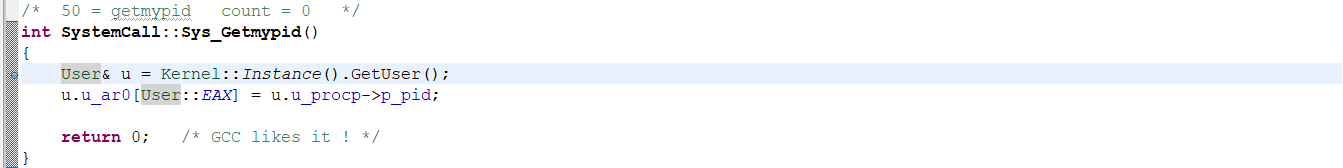
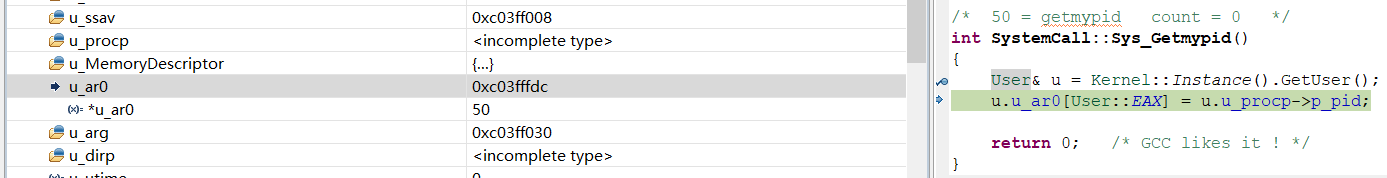
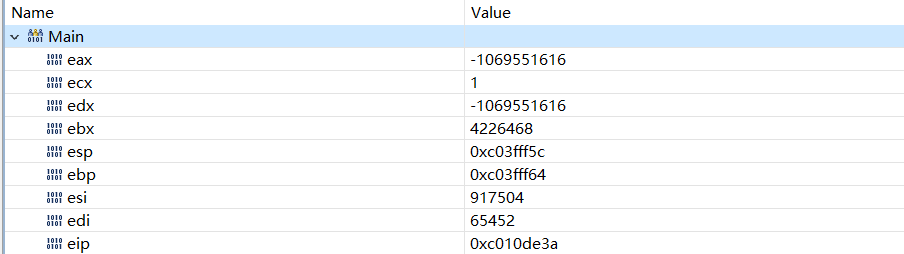


图 3-15：设置断点

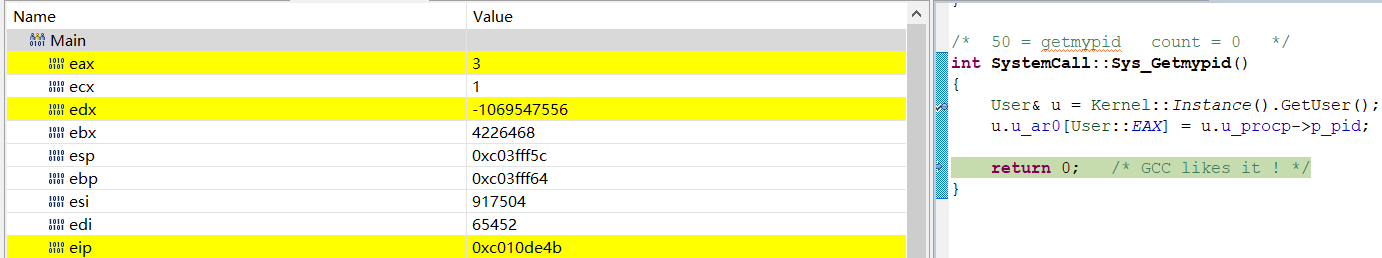
### 截图说明变量与寄存器的值的变化

断点设置好后，再次启动 UNIXV6++，在 eclipse 中启动调试，在 UNIX V6++中仍然键入如图 3-14所示的命令，程序将在断点处停下来。这个时候，可以使程序单步执行，并观察程序中各个变量和寄存器的值。

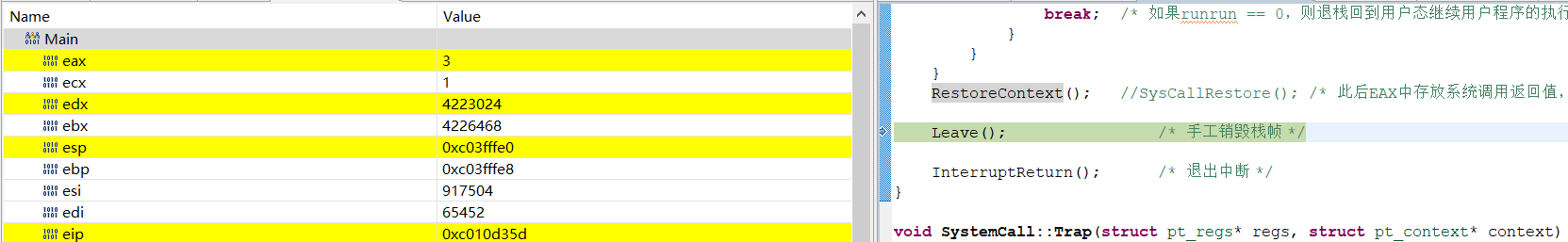




程序运行到当前位置，可以发现变量u.ar0已经显示系统调用号 50，但EAX寄存器的值仍然是乱值。



程序运行到当前位置，可以发现寄存器EAX的值已经变为系统调用子程序返回的结果3，即当前进程的ID号。



程序运行到当前位置，此后EAX中存放系统调用返回值，防止一切可能的修改。