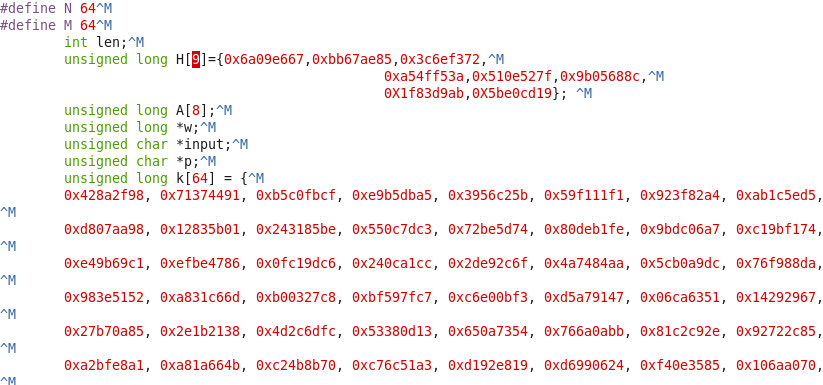
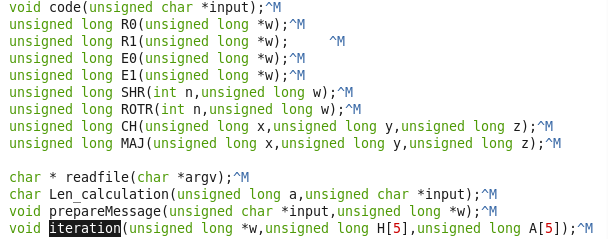
sha256算法的实现

首先定义好程序所需要的变量如：H1-H8的初始值，K数组里的64个值。还有用于接收文件首字符地址的指针\*p，和进行迭代运算512位的运算指针\*input（char类型，开辟64个char类型空间64\*8=512位），以及文件长度的整型变量len，和迭代准备值的指针\*w（char类型，开辟64个char类型空间）为了程序执行的方便和程序的需求，我将这些变量定义成全局变量。



然后声明程序运行中所需要的一些列函数。函数如下：

读取文件的函数 readfile（…），补0函数：code（…），将字符串长度转换为16进制赋值在后64位的函数Len\_calculation（…）,计算迭代运算准备值W的函数prepareMessage（…）,和迭代运算函数iteration(…) 以及SHR、ROTR、MAJ、CH、等算法的函数。



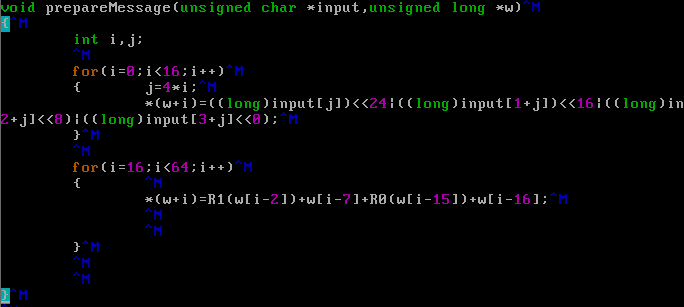
下面开始执行程序，运行主函数main。

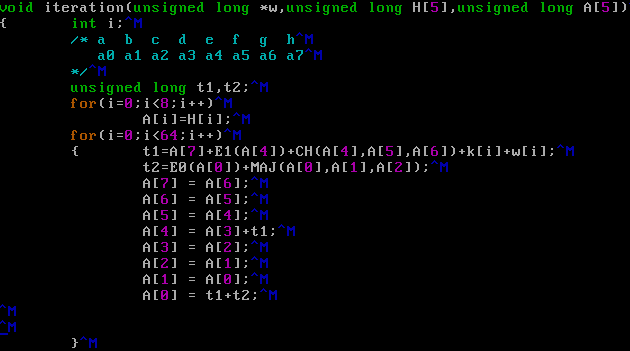


先给指针\*input，\*w开辟空间；调用readfile（…）函数读取将要加密的文件，将长度赋值个len，然后给读取文件字符的指针\*p开辟len的char类型空间，并将文件内字符传到\*p内，最后函数将这个\*p的地址返回给全局变量\*p

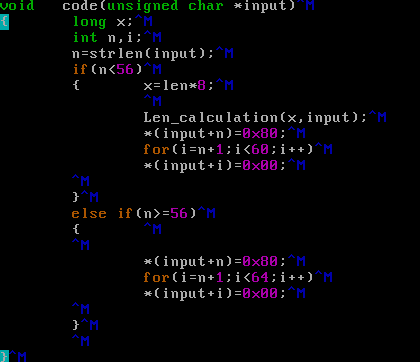


接下来将\*p的首地址传给\*input，进行运算，如果input的值过长大于512位，就先进行一系列的循环的迭代运算，调用prepareMessage(…)和iteration(…)（两函数已集成到fun（..）函数内）。如果\*input的长度小于64位时，程序跳出循环。接下来\*input则是加密字符的最后一串。就需要对其做一个判断：

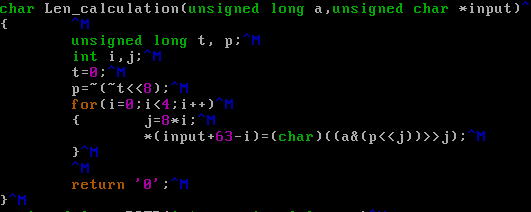




如果\*input内值小于448位，那么对input进行补0运算（调用code函数），在补0运算的同时调用Len\_calculation（…）函数将函数长度复制到input最后64位中。然后运行prepareMessage(…)和iteration(…)，做最后的迭代运算。



如果\*input内值大于448位时，对input进行补0运算（调用code函数，）然后在开辟512位，然后调用Len\_calculation（…）将文件的长度传赋值到新input的后64位中，然后在调用fun();最后迭代运输完成，输出H1-H8。



最后使用sha256sum命运进行验证：

