```
; NORESET.ASM
;
; A short TSR that patches the int 9 interrupt and intercepts the
; ctrl-alt-del keystroke sequence.
;
; Note that this code does not patch into int 2Fh (multiplex interrupt)
; nor can you remove this code from memory except by rebooting.
; If you want to be able to do these two things (as well as check for
; a previous installation), see the chapter on resident programs. Such
; code was omitted from this program because of length constraints.
;
; cseg and EndResident must occur before the standard library segments!

cseg            segment para public 'code'
OldInt9         dword   ?
cseg            ends

; Marker segment, to find the end of the resident section.

EndResident     segment para public 'Resident'
EndResident     ends

                .xlist
                include         stdlib.a
                includelib      stdlib.lib
                .list

DelScanCode     equ     53h

; Bits for the various modifier keys

CtrlBit         equ     4
AltBit          equ     8

KbdFlags        equ     <byte ptr ds:[17h]>

cseg            segment para public 'code'
                assume  ds:nothing

; SetCmd-       Sends the command byte in the AL register to the 8042
;               keyboard microcontroller chip (command register at port 64h).

SetCmd          proc    near
                push    cx
                push    ax              ;Save command value.
                cli                     ;Critical region, no ints now.

; Wait until the 8042 is done processing the current command.

                xor     cx, cx          ;Allow 65,536 times thru loop.
Wait4Empty:     in      al, 64h         ;Read keyboard status register.
                test    al, 10b         ;Input buffer full?
                loopnz  Wait4Empty      ;If so, wait until empty.

; Okay, send the command to the 8042:

                pop     ax              ;Retrieve command.
                out     64h, al
                sti                     ;Okay, ints can happen again.
                pop     cx
                ret

SetCmd          endp
```

```
; MyInt9-       Interrupt service routine for the keyboard hardware
;               interrupt. Tests to see if the user has pressed a
;               DEL key. If not, it passes control on to the original
;               int 9 handler. If so, it first checks to see if the
;               alt and ctrl keys are currently down; if not, it passes
;               control to the original handler. Otherwise it eats the
;               scan code and doesn't pass the DEL through.

MyInt9          proc    far
                push    ds
                push    ax
                push    cx

                mov     ax, 40h
                mov     ds, ax

                mov     al, 0ADh        ;Disable keyboard
                call    SetCmd
                cli                     ;Disable interrupts.
                xor     cx, cx
Wait4Data:      in      al, 64h         ;Read kbd status port.
                test    al, 10b         ;Data in buffer?
                loopz   Wait4Data       ;Wait until data available.

                in      al, 60h         ;Get keyboard data.
                cmp     al, DelScanCode ;Is it the delete key?
                jne     OrigInt9
                mov     al, KbdFlags    ;Okay, we've got DEL, is
                and     al, AltBit or CtrlBit ; ctrl+alt down too?
                cmp     al, AltBit or CtrlBit
                jne     OrigInt9

; If ctrl+alt+DEL is down, just eat the DEL code and don't pass it through.

                mov     al, 0AEh        ;Reenable the keyboard
                call    SetCmd

                mov     al, 20h         ;Send EOI (end of interrupt)
                out     20h, al         ; to the 8259A PIC.
                pop     cx
                pop     ax
                pop     ds
                iret

; If ctrl and alt aren't both down, pass DEL on to the original INT 9
; handler routine.

OrigInt9:       mov     al, 0AEh        ;Reenable the keyboard
                call    SetCmd

                pop     cx
                pop     ax
                pop     ds
                jmp     cs:OldInt9
MyInt9          endp
```

```
Main            proc
                assume  ds:cseg

                mov     ax, cseg
                mov     ds, ax

                print
                byte    "Ctrl-Alt-Del Filter",cr,lf
                byte    "Installing....",cr,lf,0

; Patch into the INT 9 interrupt vector. Note that the
; statements above have made cseg the current data segment,
; so we can store the old INT 9 value directly into
; the OldInt9 variable.

                cli                             ;Turn off interrupts!
                mov     ax, 0
                mov     es, ax
                mov     ax, es:[9*4]
                mov     word ptr OldInt9, ax
                mov     ax, es:[9*4 + 2]
                mov     word ptr OldInt9+2, ax
                mov     es:[9*4], offset MyInt9
                mov     es:[9*4+2], cs
                sti                             ;Okay, ints back on.


; We're hooked up, the only thing that remains is to terminate and
; stay resident.

                print
                byte    "Installed.",cr,lf,0

                mov     ah, 62h                 ;Get this program's PSP
                int     21h                     ; value.

                mov     dx, EndResident         ;Compute size of program.
                sub     dx, bx
                mov     ax, 3100h               ;DOS TSR command.
                int     21h
Main            endp
cseg            ends

sseg            segment para stack 'stack'
stk             db      1024 dup ("stack ")
sseg            ends

zzzzzzseg       segment para public 'zzzzzz'
LastBytes       db      16 dup (?)
zzzzzzseg       ends
                end     Main
```