

## Fire Demo ported to Linux SVGAlib – THE.UNIX.WORLD

Publicato da **Jan Wagemakers** il 09/1999

Livello **base**

### Introduzione

/\* Questo articolo è stato tratto dall'Assembly Programming Journal n° 5 \*/

```

::/ \:.....
:/___\:.....
/|   \:.....
:|   _/\:.....
:|  _ | \:.....
::\___\:.....THE.UNIX.WORLD

```

### Programmi usati

- **gcc**: compilatore C

### Iniziamo

In APJ4 there was a little nice fire demo written in DOS assembly language. I have ported this program to Linux assembly language. It is written in the AT&T-syntax (GNU assembler) and makes use of SVGAlib.

My main goal of porting this program to Linux was to show that it can be done. So, I have not optimized this program. For example, things like 'call ioperm' can also be done by making use of int 0x80; quite possibly making use of int 0x80 will make the program smaller. More information about int 0x80 is available at Konstantin Boldyshev's webpage [<http://lightning.voshod.com/asm>].

With SVGAlib you can access the screen memory directly, just like you would write to A000:0000 in a DOS asm-program. I like to thank 'paranoya' for his explanation about how to make use of SVGAlib. Anyway, enough blablabla, here is the source ;~)

```

# fire.s : fire.asm of apj 4 ported to Linux/SVGAlib
#=====
# gcc -o fire fire.s -lvga

.globl main
.type main,@function
main:
    pushl %ebp
    movl %esp,%ebp

    call vga_init # Init vga
    pushl $5
    call vga_setmode # set mode to 5 = 320x200x256
    addl $4,%esp
    pushl $0
    call vga_setpage # Point to page 0 (There is only 1 page)
    addl $4,%esp

    pushl $0x3c8 # Get IOpermission, starting from 3c8h
    pushl $2 # to 3c9h
    pushl $1 # Turn On value
    call ioperm
    addl $12,%esp

    pushl $0x60 # Get IOpermission, for 60h : keyboard
    pushl $1
    Pushl $1
    call ioperm
    addl $12,%esp

    inb $0x60,%al # Read current value of keyboard
    movb %al,key

    movw $0x3c8,%dx
    movw $0,%ax

```

```

outb %al,%dx
incw %dx

lus:
outb %al,%dx
outb %al,%dx
outb %al,%dx
incw %ax
jnz lus

movl graph_mem,%ebx

Mainloop:
movl $1280,%esi # mov si,1280 ;
movl $0x5d00,%ecx # mov ch,5dh ; y-pos, the less the faster demo
pushl %esi # push si
pushl %ecx # push cx
Sloop:
movb (%ebx,%esi),%al # lods b
incl %esi

addb (%ebx,%esi),%al # al,[si] ; pick color and
addb 320(%ebx,%esi),%al # add al,[si+320] ; pick one more and
shrb $2,%al # shr al,2

movb %al,-960(%ebx,%esi) # mov [si-960],al ; put color

loop Sloop

popl %edi # pop di
popl %ecx # pop cx

Randoml:
mulw 1(%ebx,%edi) # mul word ptr [di+1] ; 'random' routine.
incw %ax
movw %ax,(%ebx,%edi) #stosw
incl %edi
incl %edi
loop Randoml

inb $0x60,%al
cmpb key,%al
jz Mainloop

pushl $0
call exit
addl $4,%esp

movl %ebp,%esp
popl %ebp
ret

.data
key:
.byte 0

```