

ESEMPIO

```
#include <unistd.h>
#include <stdio.h>
ENVIRON è UN ARRAY DI PUNTATORI ALLE STRINGHE AMBIENTALI
extern char** environ;

void main(int argc, char **argv){
    char ** addr=environ;

    printf("process %d - environ head pointer is at address: %u\n",getpid(),(unsigned long)environ);
    fflush(stdout);

    while(*addr){
        printf("%s\n",*(addr));
        fflush(stdout);
        addr++;
    }

    execve(argv[0],argv,NULL);
//    execve(argv[0],argv,environ);
//    execve(argv[0],NULL,NULL);
//    execve("ls",NULL,environ);
//    printf("execve failed\n");
//    fflush(stdout);
}
```

Se noi chiamiamo execve di argv[0] stiamo chiamando la risostituzione di questo programma sempre con questo programma, ma lo rieseguiamo esattamente alla prima istruzione macchina, ma non stiamo clonando.

```
process 26878 - environ head pointer is at address: 183444312
GS_LIB=/home/francesco/.fonts
KDE_FULL_SESSION=true
LS_COLORS=no=00:fi=00:di=01:34:ln=00:36:pi=40:33:so=01:35:do=01:35:bd=40:33:01:cd=40:33:01:or=41:33:01:ex=00:32:*.cmd=
00:32:*.exe=01:32:*.com=01:32:*.bat=01:32:*.btm=01:32:*.dll=01:32:*.tar=00:31:*.tbz=00:31:*.tgz=00:31:*.rpm=00:31:*.deb=00:31:*.arj=00:31:*.taz=00:31:*.lzh=00:31:*.lzma=00:31:*.zip=00:31:*.zoo=00:31:*.z=00:31:*.Z=00:31:*.gz=00:31:*.bz2=00:31:*.tb2=00:31:*.tz2=00:31:*.tbz2=00:31:*.xz=00:31:*.avi=01:35:*.bmp=01:35:*.dl=01:35:*.fli=01:35:*.gif=01:35:*.gl=01:35:*.jpg=01:35:*.jpeg=01:35:*.mkv=01:35:*.mng=01:35:*.mov=01:35:*.mp4=01:35:*.mpg=01:35:*.pcx=01:35:*.pbm=01:35:*.pgm=01:35:*.png=01:35:*.ppm=01:35:*.svg=01:35:*.tga=01:35:*.tif=01:35:*.webm=01:35:*.webp=01:35:*.wmv=01:35:*.xbm=01:35:*.xcf=01:35:*.xpm=01:35:*.aiff=00:32:*.ape=00:32:*.au=00:32:*.flac=00:32:*.m4a=00:32:*.mid=00:32:*.mp3=00:32:*.mpc=00:32:*.ogg=00:32:*.voc=00:32:*.wav=00:32:*.wma=00:32:*.wv=00:32:
HOSTTYPE=x86_64
XAUTHLOCALHOSTNAME=linux-mxb5
LESSCLOSE=lessclose.sh %s %
XKEYSYMDB=/usr/X11R6/lib/X11/XKeysymDB
LANG=en_US.UTF-8
WINDOWMANAGER=/usr/bin/startkde
LESS=-M -I -R
PROFILEHOME=
DISPLAY=:0
JAVA_ROOT=/usr/lib64/jvm/jre
HOSTNAME=linux-mxb5
SHELL_SESSION_ID=0c326e2fb0314b95916311ff249146b7
CONFIG_SITE=/usr/share/site/x86_64-unknown-linux-gnu
CSHEDIT=emacs
GPG_TTY=/dev/pts/0
AUDIODRIVER=pulseaudio
lines 1-20
```

Prima stampa il PID del processo e poi stampa il puntatore alle informazioni ambientali.

Poi però questo programma rilancia se stesso con la exec ma questa volta senza parametri ambientali.

```
SESSION_MANAGER=local/linux-mxb5:@/tmp/.ICE-unix/1984,unix/linux-mxb5:/tmp/.ICE-unix/1984
CPU=x86_64
CVS_RSH=ssh
LESSOPEN=lessopen.sh %s
GTK_IM_MODULE=ibus
_=./a.out
OLDPWD=/home/francesco/git-web-site/FrancescoQuaglia.github.io/TEACHING/SISTEMI-OPERATIVI/CURRENT/SOFTWARE-BASICS
process 26899 - environ head pointer is at address: 446583224
process 26899 - environ head pointer is at address: 1057167704
process 26899 - environ head pointer is at address: 4087101944
process 26899 - environ head pointer is at address: 713851880
process 26899 - environ head pointer is at address: 4219880792
process 26899 - environ head pointer is at address: 2659336936
process 26899 - environ head pointer is at address: 166763096
process 26899 - environ head pointer is at address: 2027039752
lines 80-105
```

PERÒ IL PID DEL PROCESSO È SEMPRE LO STESSO!

Il fatto che il puntatore alle informazioni ambientali cambi ogni volta, è dovuto alla randomizzazione della stack area ogni volta.

```

#include <unistd.h>
#include <stdio.h>

extern char** environ;

void main(int argc, char **argv){
    char ** addr=environ;

    printf("process %d - environ head pointer is at address: %u\n",getpid(),(unsigned long)environ);
    fflush(stdout);

    while(*addr){
        printf("%s\n",*(addr));
        fflush(stdout);
        addr++;
    }

    // execve(argv[0], argv, NULL);
    // execve(argv[0], argv, environ);
    // execve(argv[0], NULL, NULL);
    execve("ls",NULL,environ);
    printf("execve failed\n");
    fflush(stdout);
}

```

Invece se decommentiamo la parte dedicata ad LS, notiamo che il listing non viene fatto.

```

XSESSION_IS_UP=yes
LOGNAME=francesco
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus
XDG_RUNTIME_DIR=/run/user/1000
XAUTHORITY=/home/francesco/.local/share/sddm/.Xauthority
INPUT_METHOD=ibus
JRE_HOME=/usr/lib64/jvm/java-10-openjdk-10/jre
XDG_SESSION_PATH=/org/freedesktop/DisplayManager/Session1
QT_AUTO_SCREEN_SCALE_FACTOR=0
XDG_CONFIG_DIRS=/etc/xdg
PATH=/home/francesco/bin:/usr/local/bin:/usr/bin:/bin:/usr/lib/mit/sbin
JAVA_BINDIR=/usr/lib64/jvm/jre/bin
KDE_SESSION_UID=1000
SDL_AUDIODRIVER=pulse
KDE_SESSION_VERSION=5
QT_IM_SWITCHER=imsw-multi
G_BROKEN_FILERAMES=1
HISTSIZE=1000
SESSION_MANAGER=local/linux-mxb5:@/tmp/.ICE-unix/2148,unix/linux-mxb5:/tmp/.ICE-unix/2148
CPU=x86_64
CVS_RSH=ssh
LESSOPEN=lessopen.sh %
GTK_IM_MODULE=ibus
OLDPWD=/home/francesco/git-web-site/FrancescoQuaglia.github.io/TEACHING/SISTEMI-OPERATIVI/CURRENT/SOFTWARE-EXAMPLES/PROCESSES-AND-THREADS
./a.out
francesco@linux-mxb5:~/git-web-site/FrancescoQuaglia.github.io/TEACHING/SISTEMI-OPERATIVI/CURRENT/SOFTWARE-EXAMPLES/PROCESSES-AND-THREADS/UNIX> 

```

Abbiamo lanciato il programma, abbiamo ottenuto le variabili d'ambiente in output, ma ls non è avvenuto. In qualche modo, il programma non è stato sostituito.

```

LOGNAME=francesco
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus
XDG_RUNTIME_DIR=/run/user/1000
XAUTHORITY=/home/francesco/.local/share/sddm/.Xauthority
INPUT_METHOD=ibus
JRE_HOME=/usr/lib64/jvm/java-10-openjdk-10/jre
XDG_SESSION_PATH=/org/freedesktop/DisplayManager/Session1
QT_AUTO_SCREEN_SCALE_FACTOR=0
XDG_CONFIG_DIRS=/etc/xdg
PATH=/home/francesco/bin:/usr/local/bin:/usr/bin:/bin:/usr/lib/mit/sbin
JAVA_BINDIR=/usr/lib64/jvm/jre/bin
KDE_SESSION_UID=1000
SDL_AUDIODRIVER=pulse
KDE_SESSION_VERSION=5
QT_IM_SWITCHER=imsw-multi
G_BROKEN_FILERAMES=1
HISTSIZE=1000
SESSION_MANAGER=local/linux-mxb5:@/tmp/.ICE-unix/2148,unix/linux-mxb5:/tmp/.ICE-unix/2148
CPU=x86_64
CVS_RSH=ssh
LESSOPEN=lessopen.sh %
GTK_IM_MODULE=ibus
OLDPWD=/home/francesco/git-web-site/FrancescoQuaglia.github.io/TEACHING/SISTEMI-OPERATIVI/CURRENT/SOFTWARE-EXAMPLES/PROCESSES-AND-THREADS
./a.out
execve failed
francesco@linux-mxb5:~/git-web-site/FrancescoQuaglia.github.io/TEACHING/SISTEMI-OPERATIVI/CURRENT/SOFTWARE-EXAMPLES/PROCESSES-AND-THREADS/UNIX> 

```

Innanzitutto a che directory è "ls" all'interno del file system? In questo caso abbiamo messo il nome, quindi utilizziamo la variabile d'ambiente PWD che va esattamente a cercare di lanciare questa applicazione utilizzando soltanto la directory corrente, come è registrato all'interno dell'ambiente, per questa applicazione.

Quindi questa API non è riuscita, utilizzando anche informazioni ambientali, non è riuscita ad identificare nel corretto posizionamento all'interno dell'archivio, il programma "ls".

EXECVE vede che il nome del programma non è assoluto.

```

#include <unistd.h>
#include <stdio.h>

extern char** environ;

void main(int argc, char **argv){
    char ** addr=environ;

```

```
printf("process %d - environ head pointer is at address: %u\n",getpid(),(unsigned long)environ);
fflush(stdout);

while(*addr){
    printf("%s\n",*(addr));
    fflush(stdout);
    addr++;
}

// execve(argv[0],argv,NULL);
// execve(argv[0],argv,environ);
execve(argv[0],NULL,NULL);
execve("/usr/bin/ls",NULL,environ);
printf("execve failed\n");
fflush(stdout);
}

"environment.c" 27L, 557C written
```

22,26

All

Ora invece possiamo andare a specificare la posizione corretta del programma LS.

Questo è un path name assoluto e ovviamente il kernel lancerà l'applicazione.

PERÒ ATTENZIONE: LO ABBIAMO LANCIATO CON NESSUN PARAMETRO PER IL MAIN, PERCHÉ IL PARAMETRO DOPO È NULL. IL PROGRAMMA È COMUNQUE PARTITO, MA CI SARÀ COMUNQUE UN ERRORE.

IN ARGV[0] NON HA TROVATO QUELLO CHE SI SPETTAVA, QUINDI HA ABORTITO.

Giustamente argv[0] è null ora, essendo NULL il parametro delle informazioni passate al main.

Per ovviare dobbiamo passare il pointer ad un array che definiva una serie di stringhe da passare ad ls per andare ad indicare un argv[0].

```
A NULL argv[0] was passed through an exec system call.
Aborted (core dumped)
```