X-Launcher - Il launcher universale!

Indice generale

1 - Introduzione

1.1 - Che cos'è

X-Launcher è un launcher, cioè un programma che avvia altri programmi. X-Launcher permette di modificare a piacimento le opzioni d'avvio dei programmi avviati allo scopo di renderli portatili, cioè usabili su dispositivi di memoria removibili come chiavette usb o hard disk esterni.

X-Launcher è universale, nel senso che è ampiamente configurabile e può essere usato per rendere portabile qualsiasi applicazione senza limitazioni.

L'idea è molto semplice: X-Launcher dispone di una gran numero di strumenti adatti a realizzare il suo scopo, e tutti questi strumenti possono essere configurati attraverso un file di configurazione. Non è quindi necessario conoscere alcun linguaggio di programmazione per creare un launcher che renda portatile il programma desiderato.

In sintesi si può dire che, una volta che sono chiare le modifiche da fare ad un programma per renderlo portabile, X-Lancher permette di eseguirle in modo semplice ed efficace.

1.2 - A chi è destinato

X-Launcher è uno strumento molto versatile, ma abbastanza difficile da usare. La difficoltà sta soprattutto nel fatto che bisogna aver ben chiaro quello che si deve fare per rendere un programma portatile, cosa tutt'altro che semplice!

Per usare X-Launcher non è dunque necessario conoscere linguaggi di programmazione specifici, ma è indispensabile conoscere a fondo il programma che si vuole rendere portatile, e soprattutto il modo di farlo.

E' comunque necessario avere una buona familiarità con concetti quali:

- opzioni da linea di comando;
- variabili d'ambiente;
- chiavi di registro;
- file di configurazione.

1.3 - Programmi portatili: come fare

In generale, un programma si definisce “portatile” se può essere eseguito da un supporto di memoria removibile (chiavette USB, hard disk portatili, ecc...) e in questo scenario funziona correttamente su qualsiasi computer. In particolare, ciò significa che:

1. le impostazioni devono venir salvate nel dispositivo portatile e non nel computer ospite;
2. tutti i file necessari all'esecuzione del programma devono risiedere nel dispositivo portatile.

Oltre queste due condizioni (la cui realizzazione è spesso cosa non da poco), bisogna tener

conto che spesso, nel loro funzionamento, le applicazioni fanno riferimento a percorsi assoluti, che non sono più validi cambiando il computer utilizzato.

Il principale problema da risolvere è quello di forzare il programma che si vuole rendere portatile a salvare le impostazioni nel dispositivo portatile e non nel computer ospite. Da questo punto di vista, si hanno queste tipologie di programmi:

1. programmi che salvano le impostazioni nella cartella %USERPROFILE%, oppure in %APPDATA%;

2. programmi che usano il registro di Windows. Generalmente utilizzano una sottochiave della chiave principale: “HKEY_CURRENT_USER\Software”;

3. programmi che salvano le impostazioni nella cartella del programma: questi sono “naturalmente” portatili, anche se spesso necessitano di qualche ritocco.

Purtroppo, non è sempre facile capire come funziona un programma e quindi come modificarlo. In generale, si può suggerire di fare riferimento alle risorse di supporto del programma, cioè:

- manuale d'uso;
- siti e forum di supporto;
- siti specializzati in programmi portatili.

1.3.1 - Programmi che utilizzano %USERPROFILE% o %APPDATA%

Ci sono fondamentalmente tre tecniche per forzare questo tipo di programmi a salvare le impostazioni utente in una cartella specifica al di fuori di %USERPROFILE% o %APPDATA%:

1. avviare il programma utilizzando parametri che specifichino il percorso della cartella da usare per le impostazioni. Esempi di tali programmi sono: Mozilla Firefox e Abakt.

2. avviare il programma impostando il percorso della cartella utente tramite variabili d'ambiente. Esempi di tali programmi sono: Amaya e Gaim.

3. modificare alcuni file di configurazione. Esempi di tali programmi sono: ClamWin, OpenOffice, Opera.

1.3.2 - Programmi che usano il registro di Windows

In questo caso, la procedura da seguire è un po' laboriosa, e consiste nelle fasi di seguito descritte.

Supponiamo che il programma “MyApp” salvi le sue impostazioni nella chiave di registro “HKCU\Software\MyApp”. Per poter utilizzare tali impostazioni su computer diversi, è necessario salvarle in un file di registrazione, che chiameremo “settings.reg”.

Supponiamo inoltre di trovarci nella situazione più complessa, ossia quella in cui esiste un'altra copia dello stesso programma “residente” nel computer.

Le operazioni da svolgere saranno dunque:

1. backup della chiave “HKCU\Software\MyApp”, contenente le impostazioni del programma “residente”, in un file di registrazione, che chiameremo “backup.reg”;

2. eliminazione della chiave “HKCU\Software\MyApp”: questo serve ad inizializzare l'ambiente di lavoro del nostro programma portatile;

3. caricamento delle impostazioni salvate nel file di registrazione “settings.reg”;

4. avvio e utilizzo del programma portatile;

5. chiusura del programma portatile;

6. esportazione della chiave “HKCU\Software\MyApp” nel file “settings.reg”;

7. cancellazione della chiave “HKCU\Software\MyApp”;

8. ripristino delle impostazioni del programma “residente”, precedentemente salvate nel file “backup.reg”.

Ovviamente, X-Launcher permette di eseguire tutte queste operazioni in modo completamente automatico. Esempi di tali programmi sono: Audacity, 7Zip, WackGet.

2 - Concetti base

2.1 - Il file di configurazione $ScriptIni$

Al primo avvio, X-Launcher crea il file "*X-Launcher.ini"* che dovrà essere configurato per avviare l'applicazione desiderata. Quindi, saper usare X-Launcher significa saper configurare questo file.

Il file di configurazione si trova nella stessa cartella del launcher, ed ha lo stesso nome, ma con estensione *".ini"*. Se si rinomina l'eseguibile, bisognerà quindi rinominare anche il suo file di configurazione.

Da ora in poi, il file di configurazione verrà indicato come: **$ScriptIni$**.

Una volta configurato l'.ini, è possibile ricompilare il launcher modificando l'icona e gli altri parametri per creare un launcher personalizzato.

Si ricorda che, in un file .ini, le istruzioni sono organizzate in sezioni, in questo modo:

```
[Sezione]
Chiave=Valore
```

Riguardo alle sezioni, bisogna sottolineare che:

1. il nome delle sezioni deve essere univoco: **non possono esserci due sezioni con lo stesso nome**;

2. l'ordine con cui le sezioni vengono scritte nell'.ini non ha alcuna importanza.

All'interno di ogni sezione possono essere configurate diverse chiavi, con i rispettivi valori. A seconda delle sezioni, l'ordine con cui vengono scritte le chiavi può essere importante. Sempre a seconda della sezione, possono esserci chiavi con lo stesso nome, ma con valori diversi.

Di seguito è riportato l'elenco completo delle sezioni che è possibile configurare, **nell'ordine in cui vengono interpretate da X-Launcher**:

1. [Setup]
2. [FileSystem]
3. [FileToRun]
4. [Options]
5. [SplashScreen]
6. [Environment]
7. [Functions]
8. Sezioni di riscrittura file: vengono eseguite in base all'ordine in cui appaiono nello $ScriptIni$
   - [StringReplace]
   - [WriteToFile]
   - [WriteToIni]
   - [WriteToPref]
   - [WriteToReg]
9. [RunBefore]
10. [RunAfter]

2.2 - Le variabili

La potenza di X-Launcher sta nell'uso delle variabili. Le variabili utilizzabili possono essere di tre tipi:

- variabili interne;
- macro di AutoIt;
- variabili d'ambiente.

2.2.1 - Le variabili interne

Queste variabili sono interne a X-Launcher, cioè definite nel codice sorgente. Il valore di queste variabili può essere definito dall'utente direttamente nel file di configurazione, oppure dipende dal valore di altre variabili.

Per utilizzarle, bisogna racchiudere il nome della variabile tra due simboli dollaro ($):

$internal_variable$

La lista e il significato delle variabili interne utilizzabili è riportata in appendice.

2.2.2 - Le macro

Le macro di AutoIt (il linguaggio in cui è scritto X-Launcher), sono delle variabili speciali interne al linguaggio di programmazione stesso.

Per utilizzarle, bisogna racchiudere il nome della macro tra due simboli et (@):

@autoit_macro@

La lista e il significato delle macro utilizzabili è consultabile nella guida di AutoIt, al capitolo: “Macro Reference”.

2.2.3 - Le variabili d'ambiente

Le variabili d'ambiente (*environment variable*) sono variabili poste al di fuori di ogni programma. Questo significa che le informazioni contenute in tali variabili possono essere condivise da più programmi.

In sostanza, impostando una variabile d'ambiente su X-Launcher, questa verrà recepita dal programma avviato.

X-Launcher permette di definire il valore di qualsiasi variabile d'ambiente, semplicemente configurandolo nella sezione [Environment] del file di configurazione:

```
[Environment]
VARIABILE=valore
```

Esistono inoltre variabili ambientali di sistema, ossia condivise da tutti i processi.

Per utilizzarle, bisogna racchiudere il nome della macro tra due simboli percentuale (%):

%ENVIRONMENT_VARIABLE%

Le variabili d'ambiente impostate nella sezione *[Environment]* possono essere usate in tutte le sezioni successive e nella chiave *Parameters* (sezione *[FileToRun]*), mentre le variabili ambientali di sistema possono essere ovunque.

2.3 - Variabili ausiliarie – *[Setup]*

La sezione *[Setup]* consente di impostare alcune variabili interne utili alla compilazione dello $ScriptIni$. I valori predefiniti di tali variabili sono:

```
[Setup]
AppName=MyApp
AppVer=
UserName=User
Profile=Default
Lang=%LANG%
```

Queste variabili possono essere richiamate nello $ScriptIni$ racchiudendole tra i simboli dollaro ($), quindi: $AppName$, $AppVer$, $UserName$, $Profile$, $Lang$.

Per quanto riguarda la variabile $Lang$, è da notare che se non è definita la variabile di sistema %LANG%, questa assume il valore “en” (inglese).

2.4 - Il FileSystem - *[FileSystem]*

Un concetto molto importante per capire la logica di funzionamento di X-Launcher è il *“FileSystem”*. Il FileSystem di X-Launcher non è altro che un insieme di variabili interne che descrive una struttura predefinita di cartelle di lavoro.

La variabili che definiscono il FileSystem sono riportate nella tabella seguente.

Variabile	Valore predefinito	Descrizione
$ScriptName$		Nome del launcher (es.: X-Launcher.exe --> $ScriptName$= X-Launcher)
$Root$	@ScriptDir@	Riferimento per i percorsi relativi
$Temp$	@TempDir@\$ScriptName$	Cartella temporanea del launcher
$Cache$	@TempDir@\$ScriptName$\Cache	Cartella che può essere usata come cache per programmi che ne fanno uso
$UserName$	User	Nome utente
$Home$	.\$UserName$	Cartella destinata a contenere le impostazioni dei programmi
Bin	.\Bin	Cartella destinata a contenere i file eseguibili dei programmi
Lib	.\Lib	Cartella destinata a contenere eseguibili o librerie condivise da più programmi
Doc	.\Documents	Cartella destinata ai documenti
$Backup$	.\Backups	Cartella destinata ai backup
$Download$	.\Downloads	Cartella destinata ai download

2.4.1 - Sintassi dei percorsi

Per definire i percorsi, bisogna usare la seguente sintassi:

\Path	il percorso è relativo alla cartella radice del drive di $Root$
.\Path	il percorso è relativo alla $Root$
..\Path	il percorso è relativo al livello superiore rispetto alla $Root$
..\..\Path	il percorso è relativo a due livelli superiori, e così via
X:\Path	il percorso è assoluto

X-Launcher interpreta i percorsi relativi e li trasforma in assoluti.

2.4.2 - Modifica del FileSystem

Ci sono due modi per modificare il filesystem:

- *modalità "**User**"*: le impostazioni sono scritte nello $ScriptIni$;
- *modalità "**Global**"*: le impostazioni sono scritte in un file chiamato: **X-Launcher.cfg**, posto nella stessa cartella del launcher (@ScriptDir@).

Le impostazioni al filesystem scritte nel file X-Launcher.cfg hanno effetto su tutti i launcher presenti nella cartella, mentre quelle configurate nello $ScriptIni$ hanno effetto (ovviamente) solo sul singolo launcher. Le impostazioni nello $ScriptIni$ hanno la precedenza rispetto a quelle del X-Launcher-cfg.

Le opzioni e la sintassi per modificare il filesystem sono uguali per entrambe le modalità:

```
[FileSystem]
Root=@ScriptDir@
Temp=@TempDir@\$ScriptName$
Cache=@TempDir@\$ScriptName$\Cache
Home=.\$UserName$
Bin=.\Bin
Lib=.\Lib
Doc=.\Documents
Backup=.\Backups
Download=.\Downloads
```

2.5 - Avviare un programma - *[FileToRun]*

La prima cosa da fare per configurare lo $ScriptIni$ è impostare il percorso del programma da avviare.

Nella sua forma più semplice, lo $ScriptIni$ può essere così formattato:

```
[Setup]
AppName=MyAppName

[FileToRun]
PathToExe=$Bin$\$AppName$\$AppName$.exe
Parameters=
WorkingDir=
```

La chiave *AppName*, corrispondente alla variabile **$AppName$**, indica il nome del programma da avviare, e la sua funzione è solamente quella di agevolare la compilazione dello $ScriptIni$.

La chiave *PathToExe*, corrispondente alla variabile **$PathToExe$**, indica il percorso completo dell'eseguibile da avviare. Una volta definito il percorso dell'eseguibile da avviare, sono disponibili le due variabili:

- **$ExeDir$**: cartella contenente l'eseguibile;
- **$ExeName$**: nome (completo di estensione) dell'eseguibile.

La chiave *WorkingDir*, corrispondente alla variabile **$WorkingDir$**, specifica la cartella di lavoro del programma. Se non viene specificata, la cartella di lavoro è $ExeDir$.

La chiave *Parameters* permette di impostare opzioni da linea di comando al programma. In questa chiave è possibile usare variabili d'ambiente definite nella sezione *[Environment]*.

Nota: nella chiave *PathToExe* è possibile definire semplicemente il nome dell'eseguibile da avviare, senza specificarne il percorso completo. In questo caso, l'eseguibile verrà cercato, nell'ordine, nelle seguenti cartelle:

1. $WorkingDir$ (se specificata);
2. @ScriptDir@
3. @WindowsDir@
4. @SystemDir@

Ad esempio, se si vuole lanciare Regedit, sarà sufficiente scrivere:

```
[FileToRun]
PathToExe=regedit.exe
```

2.6 - Opzioni - *[Options]*

La sezione *[Options]* consente di abilitare o disabilitare le opzioni di funzionamento del launcher.

```
[Options]
DeleteTemp=true
MultipleIstances=true
FixAppData=false
RunWait=true
ShowSplash=true
WriteLog=false
```

In questa sezione, i valori possibili per le diverse chiavi sono di tipo booleano:

- **true**: abilita l'opzione;
- **false**: disabilita l'opzione

2.6.1 - DeleteTemp

Se abilitata, elimina la cartella temporanea del launcher ($Temp$) una volta chiuso il programma avviato ($PathToExe$).

Attenzione: questa opzione è attiva solo se è contemporaneamente attiva l'opzione **RunWait**.

In caso contrario, non vi è modo per cancellare la cartella temporanea.

2.6.2 - MultipleIstances

Se disattivata, impedisce di avviare ulteriori istanze del programma avviato ($PathToExe$).

Per capire se il programma è già in esecuzione, il launcher può operare in due modi:

1. controlla l'esistenza del processo $ExeName$ (metodo predefinito);

2. controlla l'esistenza di una finestra (del programma) con un determinato titolo. Per attivare questa modalità è necessario specificare il titolo da ricercare, utilizzando la chiave "**WinGetProcess**" nella sezione *[FileToRun]*.

La seconda modalità è particolarmente indicata nel caso di applicazioni Java: infatti, quando si avvia un programma scritto in Java, in realtà viene avviato il processo "javaw.exe".

2.6.3 - FixAppData

Questa opzione permette di utilizzare programmi che salvano le proprie impostazioni in %APPDATA%.

%APPDATA% è una sottodirectory di %USERPROFILE%, e il suo nome dipende dalla lingua del sistema operativo. Ad esempio:

- in OS di lingua italiana, il suo percorso è: *"%USERPROFILE%\Dati Applicazioni"*
- in OS di lingua inglese, il suo percorso è: *"%USERPROFILE%\Application Data"*

Una volta impostato il percorso della directory %USERPROFILE%, l'opzione *FixAppData* gestisce automaticamente la directory %APPDATA%, rinominandola a seconda della lingua del sistema operativo.

Attenzione: questa opzione funziona solamente se è stata impostato il percorso della variabile %USERPROFILE%, nella sezione *[Environment]*.

2.6.4 - RunWait

Questa opzione permette di lasciare in esecuzione il launcher fino alla chiusura del programma avviato ($PathToExe$). Questo consente di:

a) eliminare la directory temporanea ($Temp$) alla chiusura;

b) eseguire le funzioni della sezione *[RunAfter]*;

c) eseguire la funzione *RegEdit* in modalità portatile.

Per capire se il programma è ancora in esecuzione, il launcher può operare in due modi:

1. controlla l'esistenza del processo $ExeName$ (metodo predefinito);

2. controlla l'esistenza di una finestra (del programma) con un determinato titolo. Per attivare questa modalità è necessario specificare il titolo da ricercare, utilizzando la chiave

"**WinGetProcess**" nella sezione *[FileToRun]*.

La seconda modalità è particolarmente indicata nel caso di applicazioni Java: infatti, quando si avvia un programma scritto in Java, in realtà viene avviato il processo "javaw.exe".

2.6.5 - ShowSplash

Abilita la visualizzazione dello splash screen all'avvio. Le proprietà dello splash screen possono essere modificate nella sezione *[SplashScreen]*.

2.6.6 - WriteLog

Abilita la creazione del file di log: *@ScriptDir@\$ScriptName$.log*

Questo file è particolarmente utile per controllare la correttezza delle impostazioni del launcher.

3 - Riferimenti alle funzioni

3.1 - Le funzioni di X-Launcher

X-Launcher possiede diverse funzioni, tutte configurabili attraverso lo $ScriptIni$, che possono essere suddivise in sei gruppi:

1. impostazione di variabili d'ambiente (l'equivalente del comando DOS: “set”);
2. operazioni su file e cartelle, eseguite prima dell'avvio del programma;
3. operazioni di riscrittura file;
4. funzioni miste, eseguite prima dell'avvio del programma;
5. funzioni miste, comprese operazioni su file e cartelle, eseguite dopo la chiusura del programma;
6. splash screen: configura lo splash screen visualizzato all'avvio.

3.1.1 - Opzioni di trasformazione delle stringhe

In molte funzioni, soprattutto in quelle di riscrittura di file, sono disponibili delle opzioni di trasformazione delle stringhe.

Tali opzioni si attivano scrivendo il simbolo della(e) trasformazione(i) da eseguire dopo il percorso, separato dal simbolo "|".

Le opzioni di trasformazione sono:

=	La stringa è presa così com'è, e nel caso si tratti di un percorso relativo, non
~	Trasforma i percorsi nel formato corto 8.3
%20	Sostituisce gli spazi nei percorsi con il simbolo "%20"
\\	Raddoppia le barre rovesciate "\"
/	Inverte le barre rovesciate
“	Racchiude la stringa tra apici

Esempio: impostiamo come $Root$ la cartella %USERPROFILE%, e creiamo la variabile d'ambiente %DESKTOP% in questo modo:

```
[FileSystem]
Root=%USERPROFILE%
[Environment]
DESKTOP=.\Desktop
```

Il percorso della cartella %USERPROFILE% è tipicamente:

%USERPROFILE%=C:\Documents and Settings\[utente]

La variabile %DESKTOP% viene interpretata come percorso relativo, quindi il suo percorso sarà:

%DESKTOP%=C:\Documents and Settings\[utente]\Desktop

Utilizzando le opzioni di trasformazione, e loro combinazioni, la variabile %DESKTOP% può essere così modificata:

```
DESKTOP=.\Desktop|\\/%20
```

%DESKTOP%=C://Documents%20and%20Settings//[utente]//Desktop

```
DESKTOP=.\Desktop|~
```

%DESKTOP%=C:\DOCUME~1\[utente]\Desktop

```
DESKTOP=.\Desktop|=
```

%DESKTOP%=.\Desktop

```
DESKTOP=.\Desktop|"/
```

%DESKTOP%="C:/Documents and Settings/[utente]/Desktop"

3.2 - Impostazione di variabili d'ambiente – *[Environment]*

La sezione *[Environment]* permette di impostare tutte le variabili d'ambiente desiderate. E' l'equivalente del comando DOS "set".

La sintassi è:

```
[Environment]
VARIABILE=valore
```

In questa sezione sono attive le opzioni di trasformazione delle stringhe.

Le variabili d'ambiente impostate nella sezione *[Environment]* possono essere usate in tutte le sezioni successive e nella chiave *Parameters* (sezione *[FileToRun]*).

3.2.1 - Variabile %PATH%

L'unico caso particolare è dato dalla variabile %PATH%, in cui - come nel corrispettivo comando DOS – è possibile specificare più percorsi di ricerca, separati da un punto e virgola (;):

```
PATH=Path1;Path2;Path3
```

Specificando %PATH%, i nuovi percorsi vengono aggiunti al precedente:

```
PATH=%PATH%;@ScriptDir@;.\Path
```

3.3 - Operazioni su file e cartelle – *[Functions]*

La sezione *[Functions]* permette di eseguire le seguenti operazioni su file e cartelle:

- *DirCopy*: copia cartelle, sottocartelle e file;
- *DirCreate*: crea cartelle vuote;
- *DirMove*: sposta una cartella, incluse le sottocartelle e i file;

- *DirRemove*: elimina una cartella
- *FileCopy*: copia uno o più file;
- *FileDelete*: elimina uno o più file;
- *FileMove*: sposta un file.

3.3.1 - DirCopy

Descrizione: copia cartelle, sottocartelle e file.

Sintassi:

```
[Functions]
DirCopy=(Sorgente)Percorso|(Destinazione)Percorso|(Opzione) o
```

La copia viene effettuata solo se la cartella di destinazione non esiste. L'opzione “o” (overwrite) permette di sovrascrivere i file e quindi effettuare la copia anche se la cartella di destinazione esiste.

3.3.2 - DirCreate

Descrizione: crea cartelle vuote.

Sintassi:

```
[Functions]
DirCreate=Percorso
```

Per creare più cartelle, è possibile usare la **sintassi compatta**:

```
DirCreate=Path1\SubPath1;SubPath2|Path2|Path3\SubPath3
```

che equivale a scrivere:

```
DirCreate=Path1\SubPath1
DirCreate=Path1\SubPath2
DirCreate=Path2
DirCreate=Path3\SubPath3
```

3.3.3 - DirMove

Descrizione: sposta una cartella, incluse le sottocartelle e i file.

Sintassi:

```
[Functions]
DirMove=(Sorgente)Percorso|(Destinazione)Percorso|(Opzione) o
```

Lo spostamento viene effettuato solo se la cartella di destinazione non esiste. L'opzione “o” (overwrite) permette di sovrascrivere i file e quindi effettuare lo spostamento anche se la cartella di destinazione esiste.

3.3.4 - DirRemove

Descrizione: elimina una cartella.

Sintassi:

```
[Functions]
DirRemove=Percorso
```

3.3.5 - FileCopy

Descrizione: copia uno o più file.

Sintassi: sono possibili due modalità a seconda che la copia coinvolga uno o più file:

- un solo file: la destinazione può essere una cartella oppure il percorso completo del file;
- più file: la destinazione è una cartella.

```
[Functions]
FileCopy=(Sorgente)Path1\file1;file2|(Destinazione)Path2\|
(Opzione) o
FileCopy=(Sorgente)Path1\file|(Destinazione)Path2\(Or)File|
(Opzione) o
```

La copia viene effettuata solo se il file di destinazione non esiste. L'opzione “o” (overwrite) permette di sovrascrivere i file e quindi effettuare la copia anche la destinazione esiste.

E' possibile usare i **caratteri jolly**, ad esempio:

```
FileCopy=Path1\*.txt|Path2\
```

3.3.6 - FileDelete

Descrizione: elimina uno o più file.

Sintassi:

```
[Functions]
FileDelete=Percorso\file
```

Per creare più file, è possibile usare la **sintassi compatta**:

```
FileDelete=Path1\file1;file2|Path2\file3
```

che equivale a scrivere:

```
FileDelete=Path1\file1
FileDelete=Path1\file2
FileDelete=Path2\file3
```

3.3.7 - FileMove

Descrizione: sposta un file.

Sintassi:

```
[Functions]
FileMove=(Sorgente)Percorso\file|(Destinazione)Percorso\file|
(Opzione) o
```

Lo spostamento viene effettuato solo se la destinazione non esiste. L'opzione “o” (overwrite) permette di sovrascrivere i file e quindi effettuare lo spostamento anche se la destinazione esiste.

3.4 - Operazioni di riscrittura file

Le operazioni di riscrittura file sono:

- *StringReplace*: sostituisce una stringa compresa tra altre due;
- *WriteToFile*: scrive righe specifiche su un file qualsiasi;
- *WriteToIni*: scrive su un file .ini standard;
- *WriteToPref*: scrive su un file di impostazione personalizzato;
- *WriteToReg*: scrive su un file di registrazione Win98/NT4 (*.reg).

Il file su cui viene effettuata l'operazione deve essere specificato nella sezione, dopo il simbolo di uguaglianza (=):

```
[Funzione=Percorso\File]
```

3.4.1 - StringReplace

Descrizione: sostituisce una stringa compresa tra altre due in una linea.

Sintassi:

```
[StringReplace=Percorso\File]
Inizio|Fine=Sostituzione
```

Dove:

- Inizio: stringa che precede la parte da sostituire;
- Fine: stringa che segue la parte da sostituire;
- Sostituzione: stringa da scrivere.

E' possibile usare le opzioni di trasformazione stringhe: = ~%20\\/"

E' possibile specificare più sostituzioni in un file.

3.4.2 - WriteToFile

Descrizione: scrive righe specifiche su un file qualsiasi.

Sintassi: nelle chiavi bisogna specificare la linea su cui scrivere, i corrispondenti valori sono il testo scritto in tali linee.

```
[WriteToFile=Percorso\File]
Line1=Questa è la prima linea
Line2=Questa è la seconda linea
...
EOF=Questa è l'ultima linea
```

Non è possibile usare le opzioni di trasformazione stringhe.

3.4.3 - WriteToIni

Descrizione: scrive su un file .ini standard.

Sintassi:

```
[WriteToIni=Percorso\File]
Sezione|Chiave=Valore
```

E' possibile usare le opzioni di trasformazione stringhe: = ~%20\\/"

3.4.4 - WriteToPref

Descrizione: scrive su un file di impostazione personalizzato.

Sintassi: per prima cosa bisogna specificare, nella chiave *Format*, il formato in cui scrivere le impostazioni nel file.

```
[WriteToPref=Percorso\File]
Format=begin[PREF]mid[VALUE]end

Preferenza1=Valore1
Preferenza2=Valore2
...
```

Dove: “begin”, “mid” e “end” vanno sostituiti con simboli opportuni, come indicato nell'esempio che segue.

E' possibile usare le opzioni di trasformazione stringhe: = ~%20\\/"

E' possibile specificare più sostituzioni in un file.

Esempio: i programmi Mozilla salvano le loro impostazioni in un file chiamato “prefs.js” (contenuto nella cartella del profilo). Aprendo questo file in editor di testo, troveremo qualcosa di simile a questo:

```
user_pref("browser.download.lastDir", "D:\\Downloads");
user_pref("browser.download.manager.alertOnEXEOpen", true);
user_pref("browser.offline", false);
user_pref("browser.preferences.privacy.selectedTabIndex", 2);
user_pref("browser.search.selectedEngine", "Google");
user_pref("browser.startup.homepage", "about:mozilla");
```

Osservando la formattazione delle linee, si può facilmente dedurre che il formato delle istruzioni è:

```
user_pref("[PREF]", [VALUE]);
```

Quindi, se volessimo impostare la cartella predefinita per il download, potremmo scrivere qualcosa del tipo:

```
[WriteToPref=%MOZ_PROFILE_PATH%\prefs.js]
Format=user_pref("[PREF]", [VALUE]);
browser.download.lastDir=\Downloads|\\”
```

Dove: %MOZ_PROFILE_PATH% indica il percorso della cartella del profilo, e deve essere impostata nella sezione *[Environment]*.

3.4.5 - WriteToReg

Descrizione: scrive su un file di registrazione Win98/NT4 (*.reg).

Sintassi: nella prima chiave, *MainKey*, bisogna impostare la chiave di registrazione principale.

```
[WriteToReg=Path\File]
MainKey=CHIAVE_PRINCIPALE

Valore1=Contenuto1
Valore2=Contenuto2
...
Sottochiave1|Valore=Contenuto
Sottochiave2|Valore=Contenuto
...
```

E' possibile usare le opzioni di trasformazione stringhe: = ~%20\\/"

Esempio: vogliamo creare un file di registrazione con le seguenti istruzioni:

```
REGEDIT4

[HKEY_CURRENT_USER\Software\Audacity\Audacity]
"WantAssociateFiles"=dword:00000000

[HKEY_CURRENT_USER\Software\Audacity\Audacity\Directories]
"TempDir"="C:\\Temp\\X-Audacity"
```

La chiave principale è ovviamente la prima, racchiusa tra parentesi quadre, quindi:

```
[WriteToReg=Audacity.reg]
MainKey=HKEY_CURRENT_USER\Software\Audacity\Audacity
"WantAssociateFiles"=dword:00000000
Directories|"TempDir"=$Temp$|\\"
```

3.5 - Funzioni miste – *[RunBefore]*

Le funzioni della sezione *[RunBefore]* vengono eseguite prima dell'avvio del programma $PathToExe$, e sono:

- *RegEdit*: installa file di registrazione *.reg;
- *RunFile*: avvia un file o esegue un comando;
- *FixDriveLetter*: riscrive un file cambiando le lettere di unità con quella di Root ($Drive$).

3.5.1 - RegEdit

Descrizione:installa file di registrazione *.reg.

Sintassi: è possibile utilizzare la sintassi compatta, specificando più file in un unico comando.

```
[RunBefore]
Regedit=Path1\File1.reg;File2.reg|Path2\File3.reg
```

Funzionamento: vi sono due possibili modalità di funzionamento:

a) **Installazione Permanente**: le chiavi contenute nel file *.reg vengono installate

permanentemente nel registro. Se tali chiavi erano già presenti nel registro, verranno sovrascritte.

b) **Installazione Temporanea**: le chiavi contenute nel file *.reg vengono installate solo per il tempo in cui il programma è in esecuzione. Se tali chiavi erano già presenti, verrà eseguito un backup nella directory temporanea, e saranno ripristinate a fine sessione. Il file *.reg sarà inoltre aggiornato con le eventuali modifiche alle chiavi operate durante l'esecuzione del programma.

Attenzione: è possibile utilizzare la modalità temporanea solo se è attiva l'opzione ***RunWait***. In questo caso, la modalità temporanea è la modalità predefinita, ed è possibile passare a quella permanente utilizzando l'opzione asterisco (*):

```
[Options]
RunWait=true

[RunBefore]
Regedit=Percorso\File1.reg|*
Regedit=Percorso\File2.reg

; File1.reg installato in modalità permanente
; File2.reg installato in modalità temporanea
```

3.5.2 - RunFile

Descrizione: avvia un file.

Sintassi:

```
[RunBefore]
RunFile=Percorso\File|Parametri
```

Esempio: è possibile avviare comandi DOS avviando l'interprete dei comandi, con le appropriate opzioni. Per esempio, il seguente comando visualizza il testo “Questo è un test” in una finestra dos:

```
RunFile=@Comspec@|/k echo "Questo è un test"
```

3.5.3 - FixDriveLetter

Descrizione: riscrive un file cambiando le lettere di unità con quella di Root ($Drive$).

Sintassi:

```
[RunBefore]
FixDriveLetter=Percorso\File
```

Opzioni: nel suo funzionamento predefinito, vengono sostituite solo le lettere di unità MAIUSCOLE. E' possibile sostituire anche le lettere di unità minuscole utilizzando l'opzione asterisco (*). E' inoltre possibile inibire la sostituzione di determinate lettere di unità utilizzando l'opzione “**skip=**”.

Ad esempio:

```
FixDriveLetter=Percorso\File|*skip=C
```

in questo modo, le lettere che fanno riferimento al disco C:\ non vengono sostituite, e, grazie all'asterisco, vengono sostituite anche le lettere di unità scritte in minuscolo.

Attenzione: nella caso di grossi file contenenti parecchi riferimenti a dischi locali, l'utilizzo di questa funzione può richiedere molto tempo per la sua esecuzione!

3.6 - Funzioni miste – *[RunAfter]*

Le funzioni della sezione *[RunAfter]* vengono eseguite dopo la chiusura del programma $PathToExe$, e sono dunque utilizzabili solo se è attiva l'opzione *RunWait.*

Le funzioni disponibili in questa sezione sono:

- *DirCopy*
- *DirMove*
- *DirRemove*
- *FileCopy*
- *FileDelete*
- *FileMove*
- *RunFile*

Tali funzioni sono le stesse già descritte nelle sezioni *[Functions]* e *[RunBefore]*.

3.7 - Configurazione dello splash screen – *[SplashScreen]*

La sezione *[SplashScreen]* serve a configurare lo splash screen visualizzato all'avvio. Sono disponibili tre opzioni:

- *Image*: imposta l'immagine da usare come splash screen. Tale immagine può essere di tipo: **Bitmap**, **GIF**, **JPEG**. Se non è impostata nessuna immagine, viene visualizzato lo splash screen predefinito.
- *Title*: imposta il titolo visualizzato nella finestra dello splash.
- *TimeOut*: imposta il tempo di visualizzazione dello splash screen (in millisecondi)

Esempio:

```
[SplashScreen]
Image=.\MyImage.jpg
Title=Test
TimeOut=3000
```

4 - Concetti avanzati

4.1 - Modificare il percorso dello $ScriptIni$

Il file di configurazione predefinito, lo $ScriptIni$ si trova nella stessa cartella del launcher, ed ha lo stesso nome, ma con estensione ".ini"(@ScriptDir@\$ScriptName$.ini).

Ci sono due modi per far usare a X-Launcher un file di configurazione diverso:

1. impostando il percorso dello $ScriptIni$ nella variabile d'ambiente %XCONFIG%
2. usando l'opzione da linea di comando: --x-launcher-config

4.1.1 - Variabile d'ambiente %XCONFIG%

La variabile d'ambiente %XCONFIG% consente di impostare il percorso dello $ScriptIni$ durante l'avvio di X-Launcher.

Ad esempio, si possono preparare diversi file di configurazione per applicazioni diverse ed avviare X-Launcher usando dei file batch ms-dos, con le seguenti, semplici, istruzioni:

```
@echo off

set XCONFIG=.\Config\Test.ini

start X-Launcher
```

In questo esempio, la cartella "Config", che si trova nella @ScriptDir@, contiene il fiel di configurazione "Test.ini".

Attenzione: nella variabile %XCONFIG% va indicato il percorso assoluto dello $ScriptIni$. Si possono usare i percorsi relativi solo nel caso in cui si usi uno script (come il batch dell'esempio) in grado di interpretare i percorsi relativi. **In questo caso, i percorsi saranno relativi allo script che avvia X-Launcher.**

4.1.2 - Opzione da linea di comando "--x-launcher-config"

Come la variabile d'ambiente %XCONFIG%, l'opzione da linea di comando "--x-launcher-config" consente di impostare il percorso dello $ScriptIni$ durante l'avvio di X-Launcher.

La sintassi dell'opzione è:

```
X-Launcher --x-launcher-config=Percorso\File.ini
```

La differenza tra usare i due modi di impostare il percorso dello $ScriptIni$ sta nel fatto che, usando l'opzione da linea di comando, **i percorsi relativi vengono interpretati da X-Launcher, e considerati relativi al launcher stesso.** Quindi, se si esegue il comando:

```
start D:\X-Launcher -x-launcher-config=.\Test.ini
```

il percorso del file di configurazione sarà: "D:\Test.ini".

5 - Appendice I – Principali variabili d'ambiente

5.1 - Variabili di sistema

Le variabili di sistema sono variabili d'ambiente impostate dal sistema operativo ed è possibile visualizzarne il valore utilizzando il comando "**set**" dal prompt dei comandi.

E' possibile modificare il valore delle variabili di sistema utilizzando la sezione *[Environment]*, e tale impostazione avrà effetto **solo** sui processi avviati dal launcher.

Le principali variabili di nostro interesse sono:

- **USERPROFILE**: directory contenente le impostazioni utente (sistemi Windows_NT). Molte applicazioni salvano le loro impostazioni in questa cartella, in modo da mantenerle differenziate per ogni utente del pc.

- **APPDATA**: è sempre una sottocartella di %USERPROFILE%, ed è usata da molte applicazioni per salvare le impostazioni differenziate per ogni utente. E' importante sottolineare che, in generale, non è possibile modificarne il valore dalla sezione *[Environment]*, o meglio, anche così facendo, la modifica non ha effetti sul programma da rendere portatile. Per i programmi che utilizzano %APPDATA% è quindi necessario impostare la directory %USERPROFILE% e attivare l'opzione *FixAppData*.

- **HOME**: nei sistemi operativi Unix, è l'equivalente della variabile %USERPROFILE%. Può essere usata anche in ambiente Windows su molti programmi di derivazione Unix/Linux, e, per questi programmi, ha la priorità rispetto alla variabile %USERPROFILE%.

- **PATH**: imposta percorsi di ricerca per file eseguibili (vedi par. 3.2.1). E' da utilizzare in tutti quei casi in cui il programma da rendere portatile necessita di conoscere la posizione di altri programmi o librerie per funzionare. E' il caso di tutti i programmi basati sulle librerie GTK, come Gimp o Gaim.

- **LANG**: imposta la lingua. In alcuni programmi, come Gimp, è l'unico modo per utilizzare una lingua differente da quella di sistema.

5.2 - Variabili particolari

Quella che segue è la lista delle variabili d'ambiente conosciute, specifiche per alcune applicazioni.

Applicazione	Variabile	Descrizione
Amaya	AMAYA_USER_HOME	Imposta la cartella Home
	AMAYA_MULTIPLE_INSTANCES=yes	Abilita istanze multiple
Mozilla	MOZ_PLUGIN_PATH	Imposta il percorso dei plugin
	MOZ_NO_REMOTE=1	Abilita istanze multiple
Gaim	GAIMHOME	Imposta la cartella Home
	GAIMLANG	Imposta la lingua

Applicazione	Variabile	Descrizione
Gimp	GIMP2_DIRECTORY	Imposta la cartella Home
GnuPG	GNUPGHOME	Imposta la cartella Home
Scite	SciTE_HOME	Imposta la cartella Home

6 - Appendice II – Macro di AutoIt

La seguente tabella raccoglie le principali macro di AutoIt che è possibile utilizzare con X-Launcher. Ulteriori informazioni sono disponibili nel manuale d'uso di AutoIt.

Macro	Desrizione
@AppDataCommonDir	Directory %APPDATA% comune
@AppDataDir	Directory %APPDATA% utente
@AutoItVersion	Numero di versione di AutoIt, ad esempio 3.0.81.0
@CommonFilesDir	Directory %CommonProgramFile%
@ComputerName	Il nome di rete del Computer.
@ComSpec	valore di %comspec%, L'interprete di comandi secondario specificato
@CR	Carriage return, Chr(13); talvolta usato per interruzioni di linea.
@CRLF	= @CR & @LF ;usato occasionalmente per interruzioni di linea.
@DesktopCommonDir	Cartella del Desktop comune
@DesktopDir	Cartella del Desktop dell'utente corrente
@DocumentsCommonDir	path alla cartella Documenti
@FavoritesCommonDir	Cartella Preferiti comune
@FavoritesDir	Vartella Preferiti dell'utente attivo
@HomeDrive	Lettera del drive contenente l'home directory dell'utente attivo.
@HomePath	Parte dell'home directory dell'utente attivo. Per ottenere il path completo, usalo in congiunzione con @HomeDrive.
@HomeShare	@HomeShare - Nome del server e della condivisione contenente l'home directory dell'utente attivo.
@HOUR	Valore delle ore dell'orologio nel formato 24-ore. Il range va da 00 a 23
@LF	Line feed, Chr(10); Tipicamente usato per interruzioni di linea.
@LogonDNSDomain	Dominio DNS di Logon.
@LogonDomain	Dominio di Logon.
@LogonServer	Logon server.
@MDAY	Giorno corrente del mese. Il range va da 01 a 31
@MIN	Valore dei minuti dell'orologio. Il range va da 00 a 59
@MON	Mese corrente. Il range va da 01 a 12
@MyDocumentsDir	path alla cartella My Documents
@ProgramFilesDir	path alla cartella Program Files
@ProgramsCommonDir	path alla cartella Programmi del menu Start
@ProgramsDir	path alla cartella Programmi dell'utente corrente (cartella del menu Start)
@ScriptDir	Directory contenente lo script in esecuzione. (Il risultato non termina con il carattere backslash)

@ScriptFullPath	Equivalente a @ScriptDir & "\" & @ScriptName
@ScriptName	Filename lungo di uno script in esecuzione.
@SEC	Valore dei secondi dell'orologio. Il range va da 00 a 59
@StartMenuCommonDir	path alla cartella del menu Start
@StartMenuDir	path al menu Start dell'utente corrente
@StartupCommonDir	path alla cartella Esecuzione automatica
@StartupDir	cartella Esecuzione automatica dell'utente corrente
@SystemDir	Cartella di sistema di Windows (System o System32)
@TAB	Carattere Tab, Chr(9)
@TempDir	Path alla cartella "temporary files".
@UserProfileDir	Path alla cartella di Profilo dell'utente corrente.
@UserName	ID dell'utente correntemente loggato.
@WDAY	Giorno numerico della settimana. Il range va da 1 a 7 e corrisponde al range da Domenica a Sabato.
@WindowsDir	Cartella di Windows
@WorkingDir	Directory di lavoro corrente/attiva. (Il risultato non contiene un carattere finale backslash)
@YDAY	Giorno corrente dell'anno. Il range va da 1 a 366 (o 365 se non è un anno bisestile)
@YEAR	Anno corrente a quattro cifre