آموزش اسکریپت‌نویسی NSIS

برای برنامه‌نویسان تازه‌کار و پیشرفته

Nullsoft Scriptable Install System

تألیف

امیرمسعود ایرانی

AMIB

amibct@gmail.com

خرداد، تیر، و مرداد۸۶

ویرایش نخست: ۲ شهریور ۸۶

منبع: راهنمای برنامه

مقدّمه

در محیط ویندوز به دلایل مختلفی از جمله استفاده‌ی برنامه‌ها از کتابخانه‌های خارجی، نیاز به ایجاد آیکون برنامه در محل‌های قابل دسترس برای کاربر، ساده‌تر کردن استفاده برای مشتریان خانگی و ... برنامه‌های نصب جایگاه ویژه‌ای در میان برنامه‌نویسان پیدا کردند. زیرا انتشار نرم‌افزار و شیوه‌ی انجام این کار می‌تواند در دیدگاه کاربر نسبت به نرم‌افزار تأثیر زیادی داشته باشد.

در حال حاضر نرم‌افزارهای متعدّد و گوناگونی برای انجام این کار توسعه یافته‌اند. شاید مشهورترین آن‌ها **InstallShield** باشد. بی‌شک امکاناتی که این نرم‌افزار در اختیار توسعه‌دهندگان قرار می‌دهد، فرآیند انتشار را بسیار سرعت می‌بخشند. با وجود همه‌ی مزایا، معایب اصلی **InstallShield** عبارتند از حجم بالای خروجی و بهای نرم‌افزار که استفاده‌ی آن را روز به روز کاهش می‌دهند.

NSIS با توجّه به توسعه‌های اخیر خود، گام به گام به سوی در انحصار گرفتن نصّاب برنامه‌ها پیش می‌رود. تأیید کننده‌ی این امر هم آمار بسیار بالای دانلود، دستیابی به چندین جایزه‌ی بین‌المللی نرم‌افزاری و رتبه‌ی بالای این نرم‌افزار در **SourceForge** (مرکز اصلی پروژه‌های متن باز اینترنت) است.

در مقاله‌ی پیش رو تلاش خواهم کرد به ساده‌ترین شکل ممکن و به گونه‌ای که برای هر برنامه‌نویس تازه کار قابل یادگیری باشد به شرح قابلیّت‌ها و آموزش استفاده از **NSIS** بپردازم.

شما نیز اگر نمی‌خواهید از کاروان رو به گسترش کاربران **NSIS** پس بمانید، این مقاله را دنبال کنید.

۱ - معرّفی

ابتدا به فهرست توانایی‌های **NSIS** می‌پردازیم:

- حجم کم سرآمد(**Stub**) – حدود ۳۵ کیلوبایت
- سازگاری با همه‌ی نسخه‌های ویندوز
- فشرده‌سازی قدرتمند و قابل انتخاب و تنظیم توسط برنامه‌نویس
- پایه‌گذاری شده بر اساس اسکریپت نویسی
- امکان ایجاد برنامه‌ی نصب با چندین زبان – پشتیبانی از زبان‌های راست به چپ مانند فارسی
- قابلیّت‌ها و بررسی‌های متعدّد برای سیستم مقصد
- امکان سفارشی کردن رابط کاربر و ایجاد تغییرات دلخواه
- پشتیبانی از **Plug-in** برای افزودن امکانات دلخواه کابر
- امکان نصب از راه شبکه و امکان ایجاد **Patch** فایل با استفاده از **Plug-in** ها
- پیش‌پردازنده‌ی قوی که امکان ایجاد برنامه‌ی نصب را به گونه‌های مختلف فراهم می‌سازد
- ساختار اسکریپت ساده و روان

همان‌گونه که باید متوجّه شده باشید **NSIS** فاقد رابط کاربر برای طرّاحی است. ولی نگران نباشید چون علاوه بر سادگی زبان **NSIS**، برنامه‌ها و ابزارهای جانبی بسیاری برای ساده‌تر کردن فرآیند برنامه‌نویسی در دسترس شماست. در این مقاله با بهترین ویرایشگرها و ابزارهای لازم آشنا خواهیم شد.

۱-۱- نصب برنامه

با توجّه به آنچه که تا کنون گفته شد انتظار بر این است که نصب **NSIS** ساده، کم حجم و سریع باشد. حجم فایل نصب فعلی فقط ۱٫۴۴ مگابایت است یعنی بر روی یک دیسکت جای می‌گیرد و در زمان نصب نیز با چند بار فشردن **Next** و انتخاب مسیر مقصد، نصب با سرعت بالا به پایان می‌رسد.

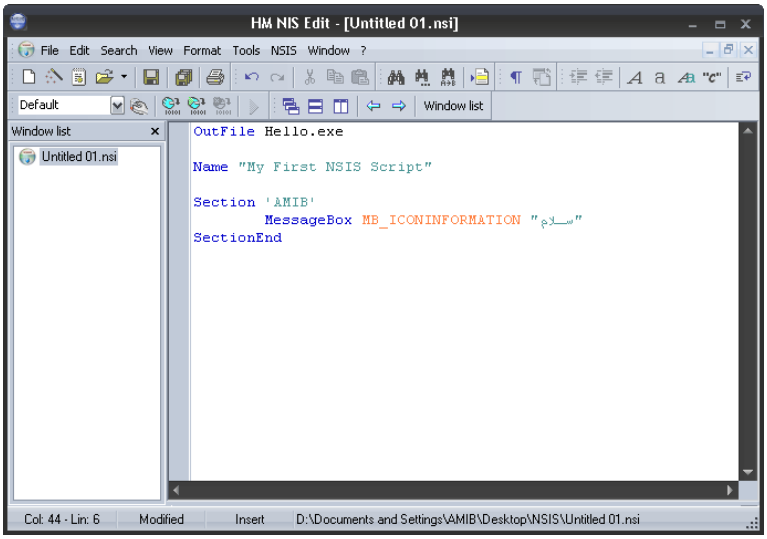
۱-۲- انتخاب ویرایشگر

به دلیل توانایی‌های **NSIS** بسیاری اقدام به نوشتن ویرایشگر برای آن نموده‌اند. در حال حاضر قویترین محیط برای توسعه‌ی برنامه‌های **NSIS** ، **Eclipse** است البته با استفاده از پلاگین **eclipsensis**.

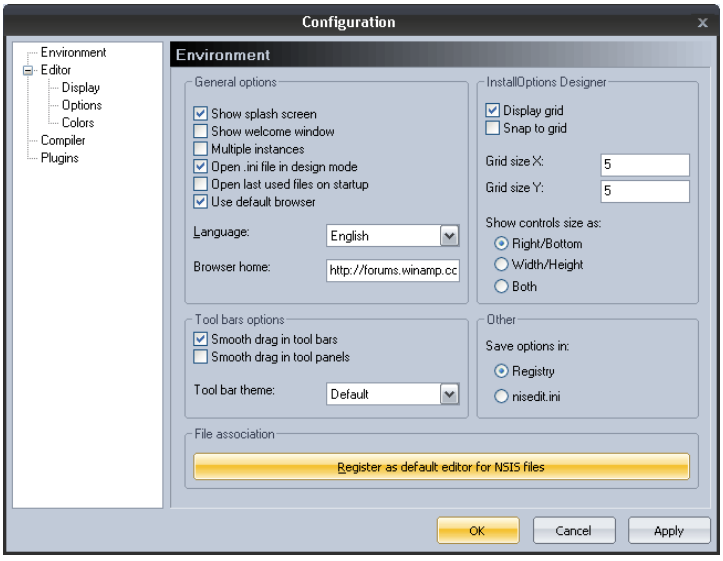
به دلیل پیچیدگی نصب **Eclipse** و پلاگین‌های آن و همچنین حجم بالا و عدم نیاز ما به توانایی‌های آن، در این مقاله از نرم‌افزارهای ساده‌تر و در عین حال با امکانات کافی استفاده خواهیم کرد.

ویرایشگر منتخب ما در این مقاله **HM NIS Edit** است. این برنامه مدّتی است که به روزرسانی نشده ولی با این حال همگی امکانات مورد نیاز یک برنامه‌نویس **NSIS** از جمله اسکریپت‌نویسی و طراحی صفحات سفارشی را پشتیبانی می‌کند.

برای دانلود **NIS Edit** به سایت **http://hmne.sourceforge.net** سر بزنید و به بخش دریافت مراجعه کنید.



پس از نصب برنامه با کمک کلید **F11** به بخش تنظیمات مراجعه کنید و دکمه‌ی مربوط به وابسته کردن فایل‌های **nsi** و **nsh** به این برنامه را فشار دهید تا بتوان از محیط **Explorer** ویندوز اسکریپت‌های **NSIS** را با این برنامه باز کرد.



۱-۳- ابزارهای کمکی

ابزارهای گوناگونی برای توسعه‌ی اسکریپت‌های **NSIS** موجود هستند که در طول مقاله و در صورت نیاز از آن‌ها استفاده خواهیم کرد

۱-۴- به NSIS خوش آمدید

پیش از شروع بهتر است با ساختار کلّی **NSIS** آشنا شوید.

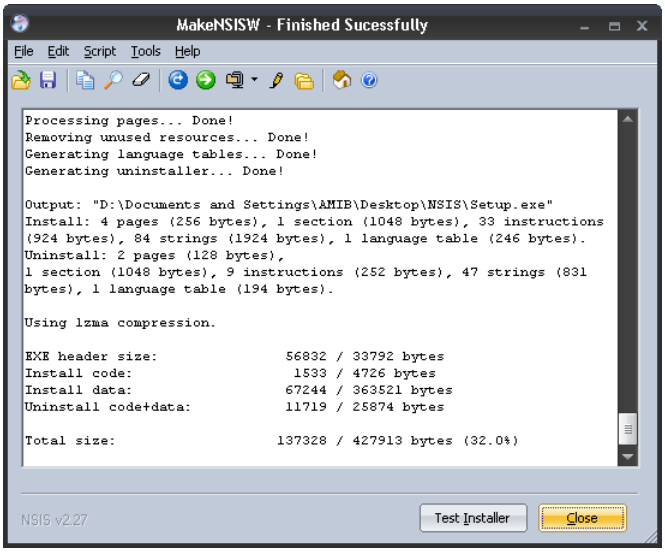
برای ایجاد یک فایل نصب باید یک فایل با پسوند **nsi** ایجاد کنید. محتویّات این فایل به کامپایلر **NSIS** می‌گوید که فایل خروجی را با چه نامی ذخیره کند، چه آیکونی داشته باشد، چه فایل‌هایی را در کجا نصب کند و ...

برای تبدیل یک فایل **nsi** به فایل اجرایی باید کامپایلر **NSIS** را فراخوانی کنید. این کار با کلیک راست روی آیکون فایل و انتخاب **Compile NSIS Script** امکان پذیر است. با انجام این کار پنجره‌ی گرافیکی **NSIS** به نمایش در خواهد آمد. این پنجره در حقیقت نقش یک رابط بین شما و کامپایلر **NSIS** را ایفا می‌کند. زیرا کامپایلر **NSIS** یک برنامه در محیط کنسول است و رابط کاربر ندارد.

در تصویر زیر می‌توانید پنجره‌ی رابط **NSIS** را مشاهده کنید.

در زمان کامپایل اطّلاعات مختلفی که می‌تواند شامل هشدار، اخطار، و آمار باشد نمایش داده می‌شوند. در انتهای اطّلاعات حجم فایل‌های ورودی و حجم برنامه‌ی نصب با یکدیگر مقایسه شده است.

علاوه بر گزینه‌ی ذکر شده در **Explorer** یک گزینه‌ی دیگر نیز در اختیار شماست که می‌توانید به کامپایلر بگویید تا انواع فشرده‌سازی‌های ممکن را آزمایش کند و کم‌حجم‌ترین خروجی را برای شما باقی بگذارد. استفاده از این گزینه در اوایل کار با **NSIS** می‌تواند به آشنا شدن شما با توانایی‌ها و سرعت هر فرمت فشرده‌سازی بینجامد.



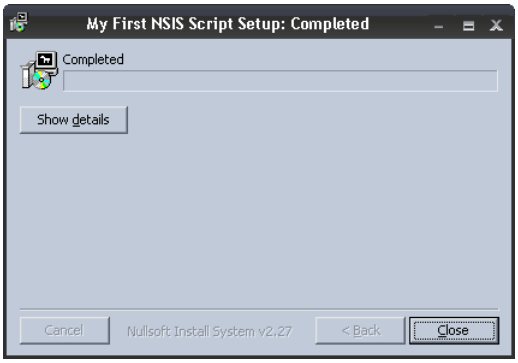
یکی دیگر از قابلیّت‌های منحصر به فرد **NSIS** که تا کنون به آن اشاره نکرده بودیم بالا نرفتن حجم برنامه‌ی نصب در صورت استفاده از **UnInstaller** است. به این دلیل که برنامه‌ی نصب توانایی اجرای دستورات حذف برنامه را نیز داراست و تنها با یک کپی‌برداری، تغییر آیکون، و اضافه کردن دستورات لازم، **UnInstaller** شما آماده است.

برای افزایش اشتیاق شما به یادگیری **NSIS**، نخستین اسکریپت خود را در همین گام خواهیم نوشت.

- **NIS Edit** یا یک ویرایشگر متن را باز کنید.
- کد ذیل را داخل ویرایشگر بنویسید.

```
OutFile Hello.exe

Name "My First NSIS Script"

Section
SectionEnd
```

- فایل را با نام **Hello.nsi** ذخیره کنید.
- اگر از **NIS Edit** استفاده می‌کنید با فشرده کلیدهای **Shift+F9** اسکریپت خود را کامپایل و اجرا کنید و در غیر این صورت از راه **Explorer** فایل مورد نظر را بیابید و با کلیک راست بر روی آن و انتخاب **Compile NSIS Script** خروجی را به وجود آورید.
- در شکل ذیل می‌توانید حاصل را مشاهده کنید.



۲- آشنایی با اسکریپت NSIS

هر زبان برنامه‌نویسی از قواعد مخصوص به خود پیروی می‌کند. در ادامه به قوانین کلّی در نوشتن برنامه‌های **NSIS** می‌پردازیم.

در **NSIS** هر خط از اسکریپت تنها یک دستور را می‌پذیرد ولی امکان اینکه یک دستور در چند خط ادامه پیدا کند وجود دارد. برای انجام این کار باید در انتهای خطوطی که قصد ادامه‌ی آنها در پایین‌تر را دارید، از نویسه‌ی ممیز وارو «\» استفاده نمایید:

```
MessageBox MB_OK|MB_ICONINFORMATION \
       "ادامه‌ي دستور در خط بعدي"
```

استفاده از توضیحات در میان دستورات و در پایان خطوط می‌تواند اشکال‌زدایی را ساده‌تر کند و باعث خوانایی بیشتر اسکریپت شما گردد. **NSIS** از توضیحات چند خطّی پشتیبانی نمی‌کند و برای اضافه کردن توضیحات باید در ابتدای آن از نویسه‌ی نقطه ویرگول «;» یا علامت شماره «**#**» استفاده کنید.

```
File /r "myDir\*.*"     ;کپي همه‌ي شاخه‌ها و فايل‌هاي موجود در شاخه
```

خصوصیّت جالب نویسه‌ی ممیّز وارو «\» عملکرد آن در انتهای خطوط توضیح است. به این معنی که می‌توانید یک خط توضیح را نیز به خط بعدی ادامه دهید.

۲-۱- رشته‌ها

برای قرار دادن رشته‌های حرفی دو گزینه‌ی نویسه‌ی نقل قول «"» و آپستروف «'» در اختیار شماست. چنانچه وجود هر یک از این دو نویسه در رشته نیاز است، می‌توانید از دیگری برای این کار استفاده کنید. و راه حلّ دیگر، استفاده از نویسه‌ی ترکیبی «"\$» یا «'\$» در میان رشته است که به ترتیب نویسه‌ی نقل قول و آپستروف را بدون تداخل با دستورات اسکریپت در رشته قرار می‌دهند.

```
Name "AMIB's Installer"                          ; AMIB's Installer
Section 'Sample Section "Named AMIB"'            ; Sample Section "Named AMIB"
MessageBox MB_ICONSTOP "Error # $\"1234$\""      ; Error #"1234"
```

موارد دیگری از نویسه‌های ترکیبی می‌توانند در میان رشته‌های حرفی قرار بگیرند مانند «**$\n**» و «**$\t**» که به ترتیب برای رفتن به خط بعدی و نویسه‌ی جدول‌بندی(**Tab**) هستند.

اگر با زبان **C** یا **PHP** آشنایی داشته باشید شاید از خود بپرسید که علامت دلار به چه دلیل در موارد بالا گنجانده شده؟ پاسخ این پرسش تعریف شدن نویسه‌های ترکیبی به عنوان متغیّر در **NSIS** است. یعنی مانند هر متغیّر دیگری برای مثال «**$PROGRAMFILES**» که مسیر پیش‌فرض نصب برنامه‌ها در سیستم مقصد را تعیین می‌کند.

استفاده از گیومه و آپستروف هنگامی اجباری است که رشته حاوی نویسه‌ی فاصله(**Space**) باشد. بنابراین چنانچه رشته‌ی شما فاقد نویسه‌ی فاصله است، می‌توانید از نویسه‌های تعریف کننده‌ی رشته چشم‌پوشی کنید.

۲-۲- متغیّرها

یکی از مهمترین نیازهای هر زبان برنامه‌نویسی متغیّر است. برای تعریف یک اسم به عنوان متغیّر، از دستور **Var** استفاده می‌کنیم.

```
Var myNumber
Var myString
```

استفاده از پارامتر «**/GLOBAL**» برای تعریف متغیّرهایی که در پهنه‌ی یک بخش یا تابع قرار می‌گیرند الزامی است. بنابراین همه‌ی متغیّرها فارغ از محلّ تعریف سراسری خواهند بود و از هر محلّی قابل دسترسی.

```
Function Dummy
      Var /GLOBAL myNumber
FunctionEnd
```

همان‌گونه که مشاهده می‌کنید در **NSIS** مانند **PHP** متغیّرها در زمان تعریف گونه نمی‌پذیرند و می‌توانند هر گونه داده‌ای را در خود ذخیره کنند.

برای استفاده از محتوای متغیّر می‌بایست از نویسه‌ی دلار «**$**» پیش از نام متغیّر استفاده نمایید. این قانون استفاده از متغیّرها در میان رشته‌های حرفی را نیز شامل می‌شود.

```
StrCpy $myString "رشته‌ي حرفي"              ; رشته‌ي حرفي
StrCpy $myString2 "myString = $myString"  ; myString = رشته‌ي حرفي
```

علاوه بر متغیّرهای کاربر، تعدادی متغیّر دیگر نیز در کامپایلر تعبیه شده‌اند که می‌توانند بنابر نیاز مورد استفاده قرار گیرند. این متغیّرها عبارتند از:

$0, $1, $2, $3, $4, $5, $6, $7, $8, $9, $R0, $R1, $R2, $R3, $R4, $R5, $R6, $R7, $R8, $R9

متغیّرهای ثابت دیگری نیز وجود دارند که با اجرای برنامه‌ی نصب مقداردهی می‌شوند مانند «**$WINDIR**» و «**$DESKTOPDIR**» که در طول اجرای برنامه می‌توانند کمکی در جهت انتخاب شاخه‌ی مناسب باشند. برای مشاهده‌ی لیست کامل متغیّرهای ثابت به راهنمای **NSIS** مراجعه کنید.

۲-۲-۱- مقادیر ثابت

در **NSIS** علاوه بر متغیّرها، امکان تعریف مقادیر ثابت نیز وجود دارد. برای تعریف این مقادیر از دستور **!define** استفاده می‌شود:

```
!define appVersion "1.54"

Caption "AMIB App ${appVersion}"    ;AMIB App 1.54 تغيير عنوان پنجره‌ي نصب به
```

همان‌گونه که مشاهده می‌کنید برای استفاده از مقدار ثابت‌های تعریف شده باید از فرمت «**${ConstName}**» استفاده کنیم.

موارد استفاده‌ی ثابت‌ها عبارتند از:

- تعریف مقادیری که در محل‌های مختلف استفاده می‌شوند. مانند نام برنامه، نسخه‌ی برنامه و ... برای جلوگیری از تکرار و خطاهای احتمالیِ حاصل از تکرار.
- قرار دادن مقادیر تاریخ، ساعت، و فرمول‌های ریاضی در ثابت مورد نظر برای ایجاد برنامه‌های نصب پویا.
- ایجاد برنامه‌های نصب هوشمند. تولید خروجی‌های مختلف فقط با تعریف یا عدم تعریف ثابت مورد نظر.

علاوه بر دستور **!define** دستورات دیگری نیز برای مدیریّتِ ثابت‌ها وجود دارند.

- دستور **!undef** برای حذف ثابتِ تعریف‌شده از لیست سراسری تعاریف استفاده می‌شود. توجّه کنید چنانچه ثابتی تعریف نشده باشد و قصد استفاده از مقدار آن را داشته باشید، مقدار خالی «» برگردانده نمی‌شود، و نام ثابت درست به همان شکل در خروجی ظاهر خواهد شد. برای مثال چنانچه **TestSym** تعریف نشده باشد، استفاده‌ی شما از **${TestSym}** خروجی «**${TestSym}**» را باعث خواهد شد.
- دستور **!ifdef** برای کامپایل شرطی بخشی از اسکریپت به کار می‌رود. به این گونه که اگر نام مقابل این دستور تعریف شده باشد، فرامین دنباله تا رسیدن به یک دستور **!endif** کامپایل خواهند شد.
- استفاده از دستور **!ifndef** عملکردی برعکس **!ifdef** دارد و چنانچه نام مورد نظر تعریف نشده باشد، فرامین دنباله کامپایل می‌شوند.
- دستور **!else** نیز همانند دیگر زبان‌های برنامه‌نویسی، عمل می‌کند و چنانچه شرط مورد نظر برقرار نباشد، عمل می‌کند.

```
!define DemoOutput "Demo"           ;تعريف ثابت مربوط به خروجي نمايشي

!ifdef DemoOutput                   ;اگر ثابتِ «خروجي نمايشي» تعريف شده بود
      File "Demo\myFile.exe"        ;فايل نمايشي را در بسته قرار بده
!else                               ;در غير اين صورت
      File "Full\myFile.exe"        ;فايل كامل را در بسته قرار بده
!endif                              ;اعلام پايان شرط
```

تعریف ثابت‌ها علاوه بر استفاده از دستور **!define** با استفاده از خط فرمان نیز امکان‌پذیر است. به این گونه که در برابر تعریف ثابت در داخل اسکریپت، از پارامتر **D/** مقابل کامپایلر **makensis.exe** استفاده می‌کنیم:

```
>makensis /Dنام ثابت=[مقدار ثابت]
```

۲-۳- توابع

همان‌گونه که تا کنون نیز مشاهده کرده‌اید **NSIS** برخلاف دیگر زبان‌ها برای جداکردن پارامترهای یک تابع به جای ویرگول «,» از فاصله‌ی خالی « » استفاده می‌کند.

```
CreateShortcut    "$STARTMENU\Programs\AMIB\Calculator.lnk" \
                  "$SYSDIR\Calc.exe"
```

توابع در **NSIS** مقدار بر نمی‌گردانند. برای برگشت دادن مقادیر باید از پشته، متغیّرهای از پیش تعریف شده، یا متغیّرهای کاربر استفاده کنید.

برای تعریف توابع از کلمه‌ی **Function** در ابتدا و سپس نام تابع استفاده می‌شود. همان‌گونه که در مورد مقدار برگشتی گفته شد، توابع کاربر پارامتر مستقیم نیز دریافت نمی‌کنند. در برابر باید از پشته یا متغیّرهای داخلی استفاده نمایید.

برای اعلام پایان توابع از **FunctionEnd** استفاده می‌کنیم.

```
Function AMIB_Function              #تعریف تابع
      # دستورات لازم
FunctionEnd                         #اعلام پایان تابع

Section AMIB
      Call AMIB_Function            #فراخوانی تابع
SectionEnd
```

توابع در **NSIS** به سه گونه تقسیم می‌شوند. گونه‌ی نخست توابع معمولی که شرح آن در بالا ذکر شد، گونه‌ی دوّم توابع **CallBack** که توسّط کاربر ایجاد می‌شوند تا بخش‌های دیگر برنامه (و نه خود کاربر)آن را فراخوانی کنند. این گونه توابع برای مطّلع ساختن کاربر از روند عملیّاتی که در اختیار کاربر نیست، ایجاد می‌شوند. توابع **CallBack** با نقطه «.» شروع می‌شوند.
برای مشاهده‌ی شرح کامل توابع **CallBack** به پیوست «الف» مراجعه کنید
و گونه‌ی سوّم توابع، آن‌هایی هستند که مربوط می‌شوند به برنامه‌ی **UnInstaller**. این گونه توابع باید پیشوند «un.» را داشته باشند. فراخوانی این توابع از داخل **Installer** و فراخوانی توابع عادّی از راه **UnInstaller** امکان‌پذیر نیست.

۲-۴- ماکروها

ماکروها از امکانات برنامه‌نویسی هستند که باعث حذف تکرار در زمان برنامه‌نویسی می‌شوند. تفاوت ماکروها با توابع در شکل اجرای آن‌هاست. توابع فقط یک بار در یک بخش از اسکریپت نوشته می‌شوند در حالی که ماکروها در هر بار فراخوانی توسّط پیش‌پردازنده به محّل مورد نظر کپی می‌شوند. می‌توان ماکروها را توابع درون خطّی نامگذاری کرد. مزیّتی که ماکروها نسبت به توابع دارند افزایش سرعت اجرای برنامه است زیرا اعمال لازم برای شروع و خاتمه‌ی کار توابع در آن‌ها صورت نمی‌پذیرد.

برای تعریف ماکروها باید از **!macro** استفاده کنید. ماکروها می‌توانند تعدادی پارامتر بگیرند. برای استفاده از پارامترهای ارسالی به ماکروها باید از **$** در ابتدای نام پارامتر استفاده کنید و نام پارامتر را در یک مجموعه «{}» محصور کنید:

```
!macro SomeMacro parm1 parm2 parm3
      DetailPrint "${parm1}"
      MessageBox MB_OK "${parm2}"
      File "${parm3}"
!macroend
```

برای استفاده از ماکروها از دستور **!insermacro** استفاده می‌شود:

```
!macro Print text
        DetailPrint "${text}"
!macroend

!insertmacro Print "متن ۱"
!insertmacro Print "متن ۲"
```

۲-۵- پشته (Stack)

پشتی محلّی است نامحدود برای ذخیره‌ی مقادیر. درست همانند متغیّرها پشته نیز گونه ندارد و هر مقداری را بدون توجّه به گونه‌ی آن می‌توان در پشته قرار داد. البته طول مجاز برای رشته‌ها ۱۰۲۴ حرف است چه در متغیّرها و چه در پشته. برای اینکه این مقدار را افزایش دهید باید از نسخه‌های سفارشی که در سایت برنامه موجود است استفاده کنید یا با توجّه به متن‌باز بودن برنامه، با ایجاد تغییرات در سورس و کامپایل مجدّد، به هدف خود برسید.

تفاوت پشته با متغیّرهای معمولی روش دسترسی به آن است. پشته مانند استوانه‌ای است که مقادیر از بالا در داخل آن گذاشته می‌شوند و در هر زمان فقط آخرین مقدار گذاشته شده قابل برداشتن است. بنابراین برای برداشتن آخرین مقدار، باید همه‌ی مقادیر از پشته برداشته شوند تا نوبت به مقدار نهایی برسد.

البته در **NSIS** دستوراتی وجود دارند که امکان دسترسی مستقیم به همه‌ی مقادیر موجود در پشته را فراهم می‌سازند. به گونه‌ی کلّی دسترسی به پشته با سه دستور **Push** برای قرار دادن مقدار، **Pop** برای برداشتن مقدار و **Exch** برای پردازش‌های پیشرفته‌تر بر روی مقادیر پشته در اختیار شما هستند.

عملکرد **Exch** به این صورت است که بر اساس پارامترهای ارسالی عملکردهای مختلفی از خود بروز می‌دهد.

- چنانچه هیچ پارامتری به این تابع ارسال نشود، دو مقدار آخر پشته را با هم تعویض می‌کند.
- چنانچه یک متغیّر به عنوان پارامتر این تابع قرار بگیرد، مقدار موجود در متغیّر با آخرین مقدار موجود در پشته عوض خواهد شد.
- اگر یک متغیّر و یک عدد «n» به این تابع ارسال شوند، مقدار متغیّر با مقدار موجود در **n** امین مقدار موجود در پشته تعویض می‌شود. توجّه کنید که چنانچه عدد ارسالی بیشتر از مقادیر موجود در **Stack** باشد یک خطای جدّی رخ می‌دهد تا برنامه‌نویس را در خطایابی اسکریپت یاری کند.

```
Push 1              ; قرار دادن عدد ۱ در پشته
Push 2              ; قرار دادن عدد ۲ در پشته
Exch                ; تعویض دو مقدار آخر در پشته یعنی ۱ و ۲
Pop $0              ; بازخواني آخرين مقدار موجود در پشته يعني ۱
```

```
StrCpy $0 1         ; قرار دادن مقدار ۱ در متغیّر $0
Push 2              ; قراردادن مقدار ۲ در پشته
Exch $0             ; تعویض مقادیر آخر پشته و متغیّر
Pop $1              ; خواندن آخرین مقدار موجود در پشته یعنی ۱
```

۲-۶-صفحات

برای انتخاب صفحات نمایش داده شده در هنگام نصب باید از دستور **Page** و در هنگام حذف از دستور **UninstPage** استفاده کنیم. این دو دستور پارامترهای مختلفی قبول می‌کنند. ترتیب استفاده از دستور **Page**، ترتیب نمایش صفحات را تعیین می‌کند.

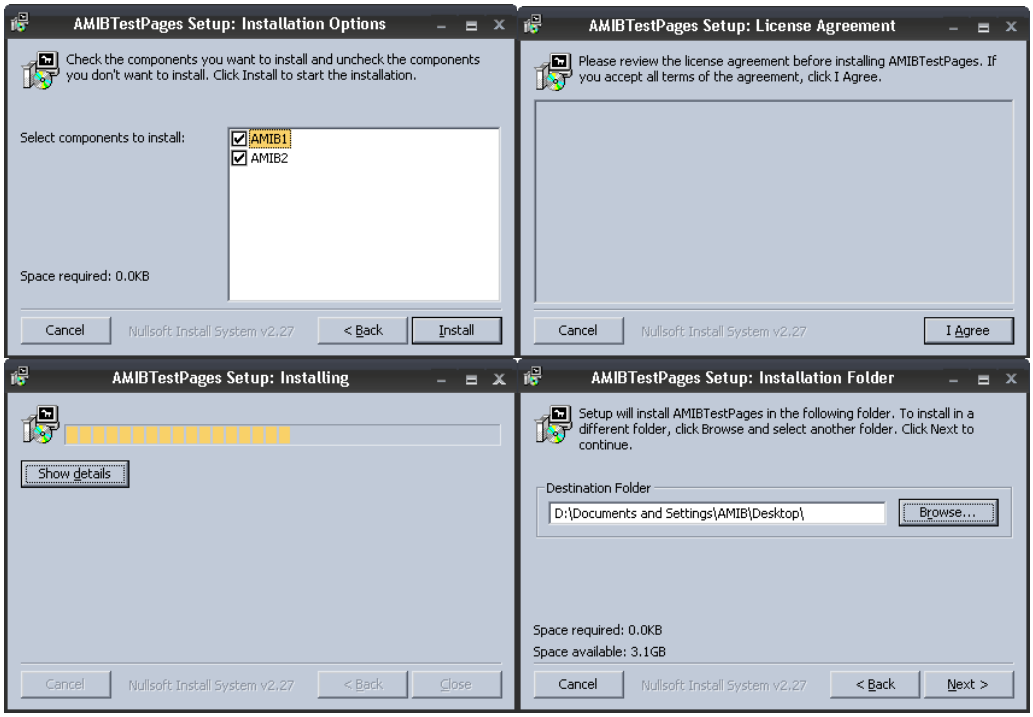
صفحاتی که **NSIS** به گونه‌ی پیش‌فرض از آن‌ها پشتیبانی می‌کند عبارتند از:

- **license** : برای نمایش مجوّز استفاده از نرم‌افزار و گرفتن تأیید کاربر برای ادامه.
- **components** : نمایش صفحه‌ی مربوط به جزئیّات نصب تا توسّط کاربر سفارشی شوند.
- **directory** : صفحه‌ی مربوط به انتخاب شاخه‌ی نصب نرم‌افزار.
- **instfiles** : نمایش صفحه‌ی اصلی برای نصب برنامه. با به نمایش در آمدن این صفحه، نصب شروع می‌شود.
- **uninstConfirm** : پرسش برای حذف نرم‌افزار.

لیست صفحاتی که به گونه‌ی معمول در هر برنامه‌ی نصبی مشاهده می‌شوند عبارتند از :

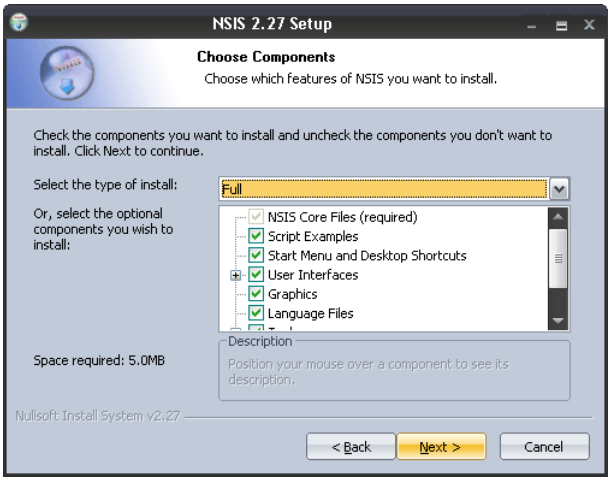
```
Page license
Page components
Page directory
Page instfiles
```

که تصاویر آن‌ها را مشاهده می‌کنید



۲-۷-بخش‌ها

استفاده‌ی بخش‌ها هنگامی است که نرم‌افزار شما می‌تواند به قسمت‌های کوچک‌تر تقسیم شود به گونه‌ای که برنامه بدون این بخش‌ها هم کارایی خود را از دست ندهد. برای مثال در برنامه‌ی نصب **NSIS** بخش‌های مختلفی وجود دارند مانند مثال‌ها، فایل‌های گرافیکی، و ... که برنامه بدون وجود آن‌ها با مشکلی برای اجرا مواجه نخواهد بود.



وجود دست کم یک بخش برای کامپایل شدن اسکریپت الزامی است. برای ایجاد بخش از دستور **Section** برای آغاز بخش و **SectionEnd** برای پایان بخش استفاده می‌شود. بخش‌ها می‌توانند مشخّصات خاصّی داشته باشند مانند ضخیم بودن عنوان، عدم امکان حذف، و ... :

- چنانچه نام بخش با خط تیره «-» آغاز شود، بی‌نام باشد، و یا نام خالی برای آن تعیین شود نمایش داده نمی‌شود.
- چنانچه پارامتر «o/» در انتهای دستور **Section** قرار بگیرد، بخش به گونه‌ی پیش‌فرض نصب نخواهد شد.
- قرار دادن علامت تعجّب «!» در ابتدای نام بخش باعث نمایش درشت عنوان آن خواهد شد.
- قرار دادن یک نام در انتهای دستور باعث می‌شود نام مورد نظر با یک مقدار عددی تعریف شود (**!define**). این مقدار عددی می‌تواند برای دسترسی به این بخش مورد استفاده قرار گیرد. به عنوان مثال برای نمایش اطّلاعات مربوط به بخش یا تغییر عنوان بخش.

```
Section "-بخش مخفي"
SectionEnd

Section                         # بخش مخفي
SectionEnd

Section "!عنوان بخش با فونت درشت"
SectionEnd

Section /o "عدم نصب پيش‌فرض - بخش انتخابي"
SectionEnd

Section "بخش معمولي" SEC_IDX   # تعريف مقدار ثابت با دستگيره‌ي بخش
SectionEnd
```

۲-۷-۱- انواع مختلف نصب

انتخاب تعداد بخش‌های زیاد می‌تواند کاربران مبتدی را سردرگم کند. برای جلوگیری از این اتّفاق باید چند گونه نصب ایجاد کنیم. برای مثال، نصب کامل، نصب کمینه، و ... .

برای ایجاد مدل‌های مختلف نصب از دستور **InstType** استفاده می‌شود. ترتیب استفاده از این دستور لیست انواع نصب را پر خواهد کرد.

```
InstType "نصب کامل"
InstType "نصب کمينه"
InstType "فقط مثال‌ها"
```

این نصب‌ها به ترتیب شماره‌های ۱ تا ۳ را به خود اختصاص خواهند داد. برای تغییر دادن بخش‌هایی که در هر یک از این نصب‌ها انتخاب‌شده خواهند بود به این شماره‌ها نیاز خواهیم داشت.

استفاده از پیشوند «un.» پیش از نام گونه‌ی نصب باعث می‌شود این گونه نصب به برنامه‌ی حذف مربوط شود و هنگام اجرای **UnInstaller** به نمایش در بیاید.

برای اینکه تعیین کنیم در هر گونه‌ی نصب چه بخش‌هایی انتخاب شده باشند در داخل پهنه‌ی بخش باید از دستور **SectionIn** استفاده کنیم و شماره‌ی انواع نصبی که در آن باید انتخاب شده باشد را به عنوان پارامتر به این دستور ارسال کنیم. چنانچه پس از شماره‌ی انواع نصب از پرچم(**Flag**) «**RO**» استفاده کنیم، از انتخاب یا عدم انتخاب این بخش توسّط کاربر جلوگیری خواهد شد.

```
SectionIn 1 2 3 RO
```

در بخش ذیل می‌توانید یک مثال از روش به‌کارگیری این دستور را مشاهده کنید.

```
Section "فایل‌های اجرایی"
	# دستورات لازم
	SectionIn 1 2		#انتخاب پیشفرض در نصب کامل و کمینه
SectionEnd

Section "مثال‌ها"
	# دستورات لازم
	SectionIn 1		#انتخاب پیشفرض در نصب کامل
SectionEnd

Section "راهنمای برنامه"
	# دستورات لازم
	SectionIn 1 3		#انتخاب پیشفرض در نصب کامل و فقط مثال‌ها
SectionEnd
```

۲-۸-دستورات پهنه‌ی سراسری

بخشی از دستورات قابل اجرا در **NSIS** مربوط به اجرا در پهنه‌ی بخش‌ها و توابع هستند و بخشی دیگر مربوط به تنظیمات اصلی نصب هستند که باید در پهنه‌ی سراسری تعریف شوند. در این قسمت به شرح تعدادی از آن‌ها می‌پردازیم.

۲-۸-۱-OutFile

وجود نام فایل خروجی و یک بخش در هر اسکریپت الزامی است. این نام باید پسوند **exe** داشته باشد و چنانچه مسیری برای آن تعیین نشود، در کنار فایل اسکریپت فعلی ایجاد خواهد شد.

```
OutFile "Setup.exe"
```

۲-۸-۲-Name

وارد کردن نام برای نصب. وارد کردن این عبارت اجباری نیست ولی چنانچه آن را وارد نکنید هشداری به شما نمایش داده می‌شود. این نام در بخش‌های مختلف برنامه‌ی نصب به کاربر نمایش داده می‌شود.

```
Name "AMIB Utilities 11.7"
```

۲-۸-۳-Caption

این دستور برای تغییر عنوان پنجره‌ی نصب به کار می‌رود. به گونه‌ی پیش‌فرض مقدار این عبارت برابر با رشته‌ی تعریف‌شده با دستور **Name** و کلمه‌ی **Setup** در انتهای آن است. ممکن است بخواهید از **NSIS** برای کاری غیر از نصب یک برنامه استفاده کنید. در این هنگام دستور **Caption** به کمک شما خواهد آمد.

```
Caption "AMIB Utilities 11.7 Security Patch"
```

۲-۸-۴-SetCompressor

در نسخه‌ی فعلی **NSIS** سه گونه فشرده‌سازی در اختیار شماست.

- **zlib** – فشرده سازی بسیار سریع و استفاده از حافظه‌ی ناچیز ۳۰۰ کیلوبایت. استفاده از این فرمت برای فایل‌هایی که پیش‌تر فشرده شده‌اند (مانند **jpg** و **mp3**) و همین گونه زمانی که سرعت نصب از اهمّیّت برخوردار است توصیه می‌شود.
- **bzip2** – فشرده‌سازی کمابیش قوی‌تر و با حافظه‌ی مورد نیاز ۴ مگابایت. سرعت این فشرده‌سازی از **zlib** کمتر است و برای فشرده‌سازی قابل قبول و نیازمند سرعت توصیه می‌گردد.

- lzma – بیشترین فشرده‌سازی ممکن. در این روش حافظه‌ی مورد نیاز در حالت پیش‌فرض ۸ مگابایت است و سرعت فشرده‌سازی و بازکردن داده‌ها کمابیش پایین است. مورد استفاده‌ی این روش هنگامی است که حجم خروجی از بیشترین اهمّیّت برخوردار است. برای مثال هنگامی که قصد انتشار خروجی از راه اینترنت را دارید. مقدار فشرده‌سازی lzma تابعی از مقدار حافظه‌ی مصرف‌شده هنگام فشرده‌سازی است. باکمک دستور SetCompressorDictSize می‌توانید مقدار حافظه‌ی مصرفی و در نتیجه قدرت فشرده‌سازی را افزایش دهید.

```
SetCompressor lzma              ;تعیین گونه‌ي فشرده‌ساز
SetCompressorDictSize 12        ;تعیین حافظه‌ي مصرفي براي افزایش قدرت
```

در صورت تمایل می‌توانید با ارسال پارامتر SOLID/ به این دستور باعث افزایش قدرت فشرده‌سازی تا حدود ۳۰ درصد شوید. ارسال این پارامتر به NSIS می‌گوید که همه‌ی فایل‌های موجود در بسته را به صورت یک فایل فرض کند و همه را با هم فشرده کند. چنانچه تعداد فایل‌ها زیاد باشد و محتوای باینری آن‌ها به یکدیگر شباهت داشته باشد، این پارامتر بیشترین تأثیرگذاری را خواهد داشت. استفاده از این دستور با افزایش هزینه همراه است و باعث کند شدن عملیّات استخراج می‌شود. به علاوه چنانچه فقط قصد استخراج یک فایل را داشته باشید، همه‌ی فایل‌های فشرده شده پیش از این فایل باید باز شوند تا قابلیّت باز شدن فایل فعلی فراهم شود.

```
SetCompressor /SOLID lzma           ;تعیین فشرده‌ساز و گونه‌ي یکپارچه
                                    ;براي افزایش بیش از پیش قدرت
```

۲-۸-۵-Icon و UninstallIcon

استفاده از این دو دستور بسیار ساده است. کافی است در مقابل این دستورات مسیر و نام فایل آیکون خود را قرار دهید تا آیکون فایل خروجی را تغییر دهید. نکته‌ی قابل ذکر لزوم مشابه بودن این دو آیکون است به این معنی که تعداد و گونه‌ی تصاویر موجود در این دو آیکون باید یکی باشد و استفاده از فایل‌های متفاوت امکان‌پذیر نیست.

```
Icon "${NSISDIR}\ Contrib\Graphics\Icons\orange-install.ico"
UninstallIcon "${NSISDIR}\ Contrib\Graphics\Icons\orange-uninstall.ico"
```

همان‌گونه که در دو دستور بالا مشاهده کردید، برای دسترسی به شاخه‌ی نصب NSIS از ثابت ${NSISDIR} استفاده شده است. این ثابت برای سهولت دسترسی به شاخه‌ی NSIS به گونه‌ی خودکار پیش از شروع کامپایل تعریف می‌شود.

با مراجعه به شاخه‌ی نصب برنامه روی دیسک خود می‌توانید بسیاری از منابع از پیش آماده برای استفاده در نصب خود بیابید.

۲-۸-۶-InstallDir

این دستور تعیین کننده‌ی مسیر پیش‌فرض نصب است. این مسیر در صفحه‌ی directory قابل تغییر است و همچنین با استفاده از تابع CallBack مربوط قابل بررسی است. (مراجعه کنید به پیوست الف)

```
InstallDir "$PROGRAMFILES\AMIB\Utilities 11.7"
```

```
InstallDir "$EXEDIR"    ;تعیین مسیر پیشفرض به مسیر اجراي برنامه‌ي نصب
```

۲-۸-۷-InstallDirRegKey

به دلایل مختلف امکان دارد مسیر نصب برنامه‌ی شما در یک شاخه از رجیستری ویندوز ذخیره شده باشد (برای مثال چنانچه این نصب برای به‌روزرسانی نسخه‌ی قبلی نرم‌افزار یا یک پلاگین برای یک نرم‌افزار دیگر باشد). در چنین شرایطی این دستور به صورت خودکار یک رشته از رجیستری را بررسی می‌کند و چنانچه مقداری داشته باشد، آن را به عنوان مسیر نصب نرم‌افزار

قرارخواهد داد. اگر رشته‌ی موردنظر نام یک فایل باشد، نام فایل حذف می‌شود و فقط بخش مسیر آن به عنوان مسیر نصب انتخاب خواهد شد.

```
InstallDirRegKey HKLM "Software\NSIS" ""  ;انتخاب مسير نصب از رجيستري
                                          ;(درصورت وجود)
```

در دستور بالا از **HKLM** به عنوان شاخه‌ی اصلی رجیستری استفاده شده است. برای دسترسی به شاخه‌های رجیستری می‌توانید از نام کامل آن‌ها یا از کوتاه‌شده‌ی آن‌ها مانند مثال بالا استفاده کنید. برای مشاهده‌ی فهرست شاخه‌های اصلی رجیستری به بخش «دستورات رجیستری» در ادامه مراجعه کنید

LicenseData-۸-۸-۲

برای تعیین فایل متنی مورد استفاده در صفحه‌ی مجوّز از این دستور استفاده می‌شود. فرمت فایل متنی می‌تواند **rtf** یا **txt** باشد. همچنین با استفاده از دستور **LicenseBkColor** می‌توانید رنگ پس‌زمینه‌ی متن مجوّز را تغییر دهید.

```
LicenseData "AMIB.rtf"
LicenseBkColor 112233   ;تعيين رنگ زمينه با مشخصات
                        ;آبي ۵۱ سبز ۳۴ قرمز ۱۷

LicenseBkColor 000000   ;تعيين رنگ زمينه به سياه
```

CRCCheck-۹-۸-۲

در ابتدای نصب و پیش از شروع کار نصب، خروجی **NSIS** با بررسی محتویّات خود تلاش می‌کند تا خطاهای احتمالی را ردیابی کند و از اجرای برنامه‌ی نصب آسیب‌دیده جلوگیری کنید.

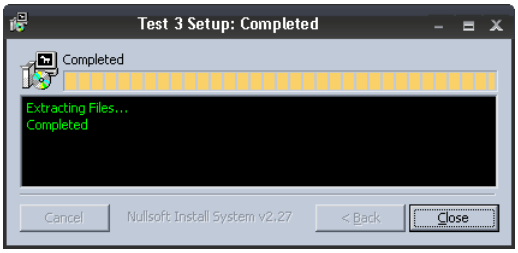
می‌توانید با خاموش کردن این بررسی، سرعت عملکرد نصب را تا حدودی بالا ببرید ولی با این کار امکان ایجاد خطاهای غیر قابل پیش‌بینی را افزایش خواهید داد.

برای غیر فعّال کردن این بررسی، دستور فوق را با پارامتر **off** فراخوانی کنید. دیگر پارامترهای قابل قبول برای این دستور **on** و **force** هستند که به ترتیب برای فعّال کردن بررسی (پیش‌فرض) و بررسی اجباری (چنانچه کاربر با ارسال پارامتر **/NCRC** به نصب، قصد غیر فعّال کردن آن را داشت) به کار می‌روند.

ChangeUI-۱۰-۸-۲

ظاهر خروجی‌هایی که تا کنون ایجاد کردیم بسیار ساده و قدیمی به نظر می‌رسند. **NSIS** امکان ایجاد تغییرات در شکل و اندازه‌ی پنجره‌ها را با استفاده از دستور **ChangeUI** فراهم می‌سازد. این دستور دو پارامتر می‌گیرد که تعیین کننده‌ی صفحه‌ی مورد نظر برای ایجاد تغییر و فایل حاوی اطّلاعات پنجره است.

```
ChangeUI all "${NSISDIR}\Contrib\UIs\sdbarker_tiny.exe"
```



با مراجعه به شاخه‌ی ذکر شده در مثال بالا، چندین نمونه‌ی آماده از رابط‌های کاربر آماده را مشاهده خواهید کرد. شما نیز می‌توانید با استفاده از برنامه‌ای مانند **Resource Hacker** رابط دلخواه خود را ایجاد نمایید. برای این کار از یکی از فایل‌های موجود در شاخه‌ی فوق یک کپی بگیرید و سپس تغییرات لازم را بر روی آن ایجاد نمایید. برای جلوگیری از ایجاد مشکلات احتمالی از حذف اشیای موجود خودداری کنید و در صورت نیاز صفت **Visible** آن شیء را خاموش نمایید.

برای اشاره به صفحه‌ی مورد نظر باید از ثابت‌های قابل قبول این دستور استفاده نمایید.

- IDD_LICENSE
- IDD_DIR
- IDD_SELCOM
- IDD_INST
- IDD_INSTFILES
- IDD_UNINST
- IDD_VERIFY

و یا مانند مثال از ثابت all برای تغییر دادن همه‌ی صفحات.

BrandingText-۱۱-۸-۲

برای تغییر متنی که در بخش پایین نصب به نمایش درمی‌آید از این دستور استفاده می‌کنیم. این متن به صورت پیش‌فرض برابر **Nullsoft Install System** است. توصیه می‌شود تا جایی که می‌توانید این متن را تغییر ندهید تا از **NSIS** که به صورت رایگان در اختیار شماست، حمایت کرده باشید.

برای قراردادن مقدار پیش‌فرض از یک رشته‌ی خالی و برای حذف کردن این عبارت، از یک فاصله‌ی خالی « » به عنوان پارامتر این دستور استفاده کنید.

```
BrandingText "این متن را عوض نکنید"      # تغییر متن

BrandingText ""                          # متن پیش‌فرض

BrandingText " "                         # متن خالی
```

AutoCloseWindow-۱۲-۸-۲

استفاده از این دستور با پارامتر **true** باعث بسته شدن خودکار پنجره‌ی نصب پس از پایان عملیّات می‌شود. درصورتی که از این دستور استفاده نکنید، به صورت پیش‌فرض بسته‌شدن پنجره‌ی نصب در اختیار کاربر است. همچنین می‌توانید با دستور **Quit** در داخل بخش‌ها یا توابع در هر زمانی به نصب خاتمه دهید.

```
AutoCloseWindow true
```

SilentInstall-۱۳-۸-۲

با فراخوانی این دستور و ارسال پارامتر **silent** به آن، همه‌ی مراحل نصب غیر از پیام‌هایی که شما آن‌ها را نمایش می‌دهید به صورت مخفی انجام خواهد شد و برای همگی بخش‌ها از مقادیر پیش‌فرض استفاده خواهد شد.
چنانچه قصد ساخت نصب بی‌صدا(**Silent**) را دارید، فراموش نکنید که از دستوراتی مانند **MessageBox** در اسکریپت خود استفاده نکنید و یا در صورت استفاده، با کمک دستورات شرطیِ مربوط به این کار از نمایش آن‌ها جلوگیری به عمل آورید.

```
SilentInstall silent

Section
      # دستورات نصب
      IfSilent +2       #پرش به دو دستور جلوتر در صورت نصب بی‌صدا
           MessageBox MB_OK "نصب به پایان رسید"
SectionEnd
```

SetFont-۱۴-۸-۲

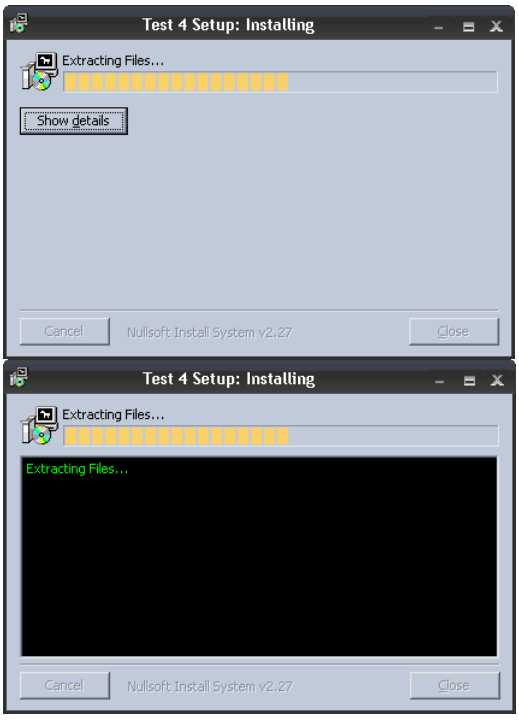
برای تغییر دادن فونت استفاده شده در همه‌ی صفحات از این دستور استفاده می‌کنیم. این دستور دو پارامتر نام و اندازه‌ی فونت را دریافت می‌کند.

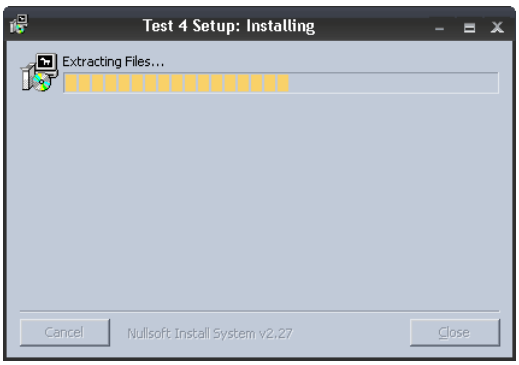
```
SetFont "Tahoma" 10
```

ShowInstDetails-۱۵-۸-۲

به گونه‌ی پیش‌فرض عملیّاتی که در حین نصب انجام می‌شود، توسّط کاربر قابل مشاهده نیست و کاربر در صورت نیاز می‌تواند آن‌ها را مشاهده کند. برای تغییر این عملکرد، باید این دستور را صدا بزنید و یکی از پارامترهای زیر را به آن ارسال نمایید.

- **hide** – حالت پیش‌فرض – عملیّات مخفی و قابل نمایش توسّط کاربر.
- **show** – عملیّات قابل مشاهده.
- **nevershow** – عملیّات مخفی و غیر قابل نمایش توسّط کاربر.





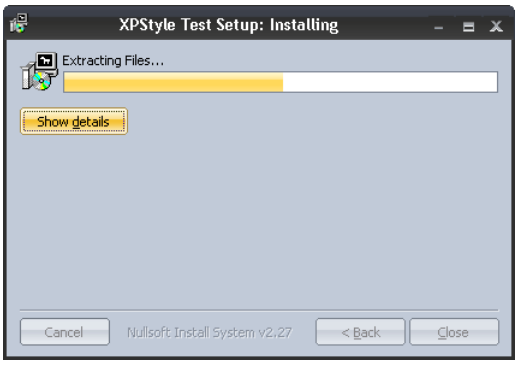
توجّه کنید که فراخوانی این دستور فقط شامل نمایش لیست عملکردها می‌شود و مانع نمایش فعّالیّت در حال اجرا نمی‌شود. برای جلوگیری از نمایش هرگونه اطّلاعات باید از دستورات قابل اجرا در بخش‌ها استفاده کنید که در ادامه به شرح آن‌ها خواهیم پرداخت.

XPStyle-۱۶-۸-۲

همگی نصب‌هایی که تا کنون مشاهده کردیم ظاهری شبیه به برنامه‌های موجود در ویندوز ۹۸ داشتند. برای اینکه جلوه‌های بصری ویندوز **XP** بر روی اشیای موجود اعمال شود باید پارامتر **on** را به دستور فوق ارسال کنیم.

```
XPStyle on
```

فراخوانی این دستور ظاهر برنامه‌ی حذف را نیز شامل می‌شود.



۱۷-۸-۲- مرور آموخته‌ها تا کنون

برای آشنایی بیشتر با عملکرد دستورات پهنه‌ی سراسری که تا کنون آموختیم مثال زیر و توضیحات آن را با دقّت مطالعه کنید.

```
!define AppName "AMIB Tools"          #تعريف نام و نسخه‌ي برنامه در ثابت
!define AppVer "2.25"                 #براي استفاده در دستورات ديگر

OutFile "${AppName} Setup.exe"        #تعيين نام خروجي با استفاده از ثابت

SetCompressor /SOLID lzma             #تعيين گونه‌ي فشرده‌سازي و
SetCompressorDictSize 6               #حافظه‌ي مورد استفاده

Icon "AMIB.ico"                       #تعيين آيكون براي خروجي
Name "${AppName} ${AppVer}"           #تعيين نام برنامه با كمك ثابت‌ها

Caption "${AppName} ${AppVer} Installer"  #تعيين صريح عنوان پنجره‌ي نصب
InstallDir "$PROGRAMFILES\${AppName}"     #تعيين مسير پيشفرض نصب
```

```
ShowInstDetails nevershow           #عدم نمايش فهرست اعمال انجام شده

Page license                        #تعيين ترتيب نمايش صفحات
Page components
Page directory
page instfiles

LicenseData 'AMIB.rtf'              #فايل حاوي متن مجوّز استفاده از برنامه
AutoCloseWindow true                #بستن خودكار پنجره‌ي نصب پس از پايان
XPStyle on                          #استفاده‌ي نصب از جلوه‌هاي تصويري XP

Section "بخش آزمايشي با عنوان درشت!"
      #عمليّات مورد نياز در بخش نخست
SectionEnd

Section /o "بخش آزمايشي ۲ انتخابي توسّط كاربر"
      #انجام كارهاي لازم در بخش ۲
SectionEnd

Section "-بخش مخفي"
      #انجام كارهاي لازم در بخش ۳
SectionEnd
```

۲-۸-۱۸-دستورات کاربردی کامپایلر

کامپایلر **NSIS** مانند کامپایلرهای دیگر علاوه بر دستوراتی که خروجی را تحت تأثیر قرار می‌دهند، از تعدادی دستور پیش‌پردازنده نیز پشتیبانی می‌کند. مورد استفاده‌ی این دستورات انجام عملیّاتی است که انجام آن‌ها با کمک امکانات داخلی **NSIS** امکان‌پذیر نیست. مانند فشرده‌کردن **Stub** خروجی، تغییر دادن یک فایل (مثلا فایل مجوّز) برای ساخت نصب‌های متفاوت در هر کامپایل، و ...

در این قسمت سعی می‌کنیم به گونه‌ی سطحی با این دستورات و کاربرد آن‌ها آشنا شویم. برای کسب اطّلاعات بیشتر به راهنمای برنامه مراجعه کنید.

– **!cd** – این دستور مانند محیط داس وظیفه‌ی تغییر مسیر فعلی را بر عهده دارد. در شروع کامپایل مسیر فعلی معادل مسیر فایل اسکریپت قرار می‌گیرد.

```
!cd "C:\AMIB\Utils"
```

– **!include** – برای ادغام محتویّات یک اسکریپت خارجی در اسکریپت فعلی از این دستور استفاده می‌کنیم. توجّه کنید که با استفاده از این دستور مسیر فعلی همچنان تغییر نخواهد کرد و چنانچه فایل ادغام شده حاوی دستورات وابسته به مسیر باشد، ممکن است با پیام خطا مواجه شوید.

با کمک پارامتر **NONFATAL**/ می‌توانید از ایجاد پیام خطا در صورت عدم وجود فایل **include** جلوگیری کنید.

```
!include "myFuncs.nsh"
!include /NONFATAL "Funcs2.nsh"      #این فایل می‌تواند موجود نباشد
```

– **!addincludedir** – برای اضافه کردن یک شاخه به فهرست شاخه‌های مورد جستجوی **include** ها از این دستور استفاده می‌کنیم. به صورت پیش‌فرض فقط شاخه‌ی **include** های **NSIS** در این فهرست قرار دارد. شاخه‌های موجود در این فهرست به ترتیب جستجو می‌شوند و چنانچه فایل **include** در هیچ یک از این شاخه‌ها موجود نباشد پیام خطایی ظاهر خواهد شد.

```
!addincludedir "C:\AMIB\NSIS_Includes"
```

– **!addplugindir** – در حالت پیش‌فرض فقط شاخه‌ی **Plugins** موجود در شاخه‌ی نصب برنامه برای پلاگین‌ها مورد جستجو قرار می‌گیرد. برای تغییر دادن این رفتار، از دستور فوق استفاده می‌کنیم.

```
!addplugindir myFolder
myPlugin::someFunction          #فراخوانی یک تابع از یک پلاگین
```

– **appendfile** – اضافه کردن رشته‌ی متنی به انتهای یک فایل.

```
!tempfile FILE                          #ایجاد یک فایل موقّت و «تعریف» نام آن
!appendfile "${FILE}" "XPStyle on$\n"    #اضافه کردن رشته به انتهای فایل
!appendfile "${FILE}" "Name 'test'$\n"   #اضافه کردن یک خط دیگر
!include "${FILE}"                       #ادغام فایل موقّت در اسکریپت فعلی
!delfile "${FILE}"                       #حذف فایل موقّت
!undef FILE                              #حذف «تعریف» نام فایل موقّت
```

– **!delfile** – حذف یک فایل از روی دیسک. (به مثال موجود در دستور **!appendfile** مراجعه کنید)

– **!execute** – فراخوانی یک فایل اجرایی و برگشت فوری به ادامه‌ی کامپایل اسکریپت.

```
!execute '"%WINDIR%\notepad.exe" "${NSISDIR}\license.txt"'
```

– **!packhdr** – برای فشرده‌کردن فایل اجرایی خروجی از این دستور استفاده می‌شود. با توجّه به اینکه پیش از کامپایل تغییراتی در فایل اجرایی ایجاد می‌شود، امکان فشرده‌کردن فایل **stub** موجود در شاخه‌ی **Stubs** وجود ندارد. بنابراین **NSIS** پس از ایجاد تغییرات لازم در فایل اجرایی چنانچه این دستور را فراخوانی کنید، خروجی را با توجّه به فشرده‌سازی که شما انتخاب کنید فشرده می‌کند تا چند کیلوبایتی از حجم خروجی شما کاسته شود.
این دستور دو پارامتر می‌پذیرد که پارامتر نخست نام فایلی موقّت و پارامتر دوّم فرمان لازم برای فشرده‌کردن همان فایل خواهد بود:

```
!packhdr "$%TEMP%\tmpstub.tmp" '"C:\UPX\upx.exe" "$%TEMP%\tmpstub.tmp"'
```

با توجّه به دستور بالا، **NSIS** فایل اجرایی **Stub** را با نام **tmpstub** در شاخه‌ی **Temp** شما ایجاد می‌کند، فشرده‌ساز **UPX** را با پارامتر مورد نظر فراخوانی می‌کند و پس از فشرده شدن توسّط **UPX** آن را در ابتدای خروجی قرار خواهد داد.

– **!system** – فراخوانی یک فایل اجرایی و مقایسه‌ی مقدار بازگشتی وظیفه‌ی این دستور است.
با کمک این دستور می‌توانید دستورات قابل اجرا در محیط **Batch File** را نیز اجرا کنید و نیز می‌توانید با کمک مقایسه‌گرهای =، **<>**، **<**، و **>** مقدار خروجی را با مقدار دلخواه مقایسه کنید. چنانچه نتیجه‌ی مقایسه نادرست(**False**) باشد، فرآیند کامپایل متوقّف خواهد شد.

```
!system "echo Hello > myFile.txt"    #(نوشتن متن داخل فايل) اجراي فرامين داس
```

```
!system "c:\UPX\upx.exe" <> 0        #اجراي برنامه و مقايسه‌ي خروجي آن
                                     #با صفر - چنانچه خروجي غير از صفر
                                     #باشد، كامپايل متوقّف مي‌شود
```

!tempfile – فراخوانی این دستور باعث ایجاد یک فایل موقّتی جدید می‌شود. عبارتی که در مقابل این دستور به کار رود به عنوان یک ثابت تعریف می‌شود و مسیر فایل مورد نظر را در خود جای خواهد داد.

```
!tempfile myTempFile            #ايجاد فايل موقّت و تعريف نام آن
!delfile "${mytempFile}"        #حذف فايل موقّت از ديسك
!undef myTempFile               #حذف تعريف نام فايل موقّت
```

```
!tempfile tmpStub               #ايجاد فايل موقّت

#فشرده كردن سرآمد فايل خروجي با كمك گرفتن از نام فايل موقّت ايجاد شده
!packhdr "${tmpStub}" '"C:\UPX\upx.exe" "${tmpStub}"'
```

۲-۹-دستورات پهنه‌ی توابع و بخش‌ها

۲-۹-۱-SetOutPath

یکی از مهم‌ترین دستورات که می‌توان گفت استفاده از آن در همه‌ی نصب‌ها مورد نیاز است، دستور **SetOutPath** می‌باشد. عملکرد این دستور در تغییر مسیر فعلی برای بسیاری از عملیّات از جمله استخراج فایل است. چنانچه مسیر خروجی را تعیین نکنید، مسیر اجرای فایل نصب به عنوان مسیر خروجی تعیین می‌شود. با اجرای این دستور شاخه‌ی مورد نظر در صورت موجود نبودن ایجاد خواهد شد.

```
Section
      SetOutPath $INSTDIR
SectionEnd
```

در مثال بالا از متغیّر **$INSTDIR** به عنوان پارامتر ارسالی به **SetOutPath** استفاده شد. این متغیّر در ابتدای نصب مقدار پیش‌فرضی معادل پارامتر ارسالی به دستور **InstallDir** یا رشته‌ی موجود در رجیستری هنگام استفاده از دستور **InstallDirRegKey** یا مقدار خالی خواهد گرفت.

در صفحه‌ی **directory** کاربر می‌تواند این مقدار را تغییر دهد. بنابراین این متغیّر همیشه محلّ نهایی برای نصب را در خود نگهداری می‌کند. همچنین در هنگام حذف، این متغیّر مسیر فعلی نصب را در خود نگهداری می‌کند و با کمک آن می‌توانید فایل‌های برنامه را از سیستم مقصد پاکسازی کنید.

۲-۹-۲-File

دستور **File** وظیفه‌ی بسته‌بندی فایل در زمان کامپایل و استخراج در زمان اجرا را بر عهده دارد. با فراخوانی این دستور و تعیین نام فایل به عنوان پارامتر، فایل‌های مورد نظر در صف فشرده‌شدن قرار می‌گیرند. ترتیب قرار گرفتن فایل‌ها در بسته متناسب با فراخوانی این دستور خواهد بود.

صدا زدن این دستور اشکال گوناگونی دارد. ساده ترین گونه‌ی آن تعیین نام یک فایل در جلوی این دستور است. برای استخراج دسته‌ای از فایل‌ها می‌توان از عملگرهای * و ? به عنوان نام فایل استفاده کرد.

```
File "C:\Windows\*.bmp"         #قرار دادن فايل‌هاي تصويري در بسته
```

مسیری که در دستور بالا مشاهده می‌کنید مسیر قرارگیری فایل‌ها در کامپیوتر مبدأ است و مسیر استخراج فایل متناسب با فراخوانی دستور **SetOutPath** تعیین می‌شود.

```
SetOutPath "C:\MyApp"           #تعيين مسير خروجي براي استخراج فايل
File "AppFiles\*.*"             #قرار دادن همه‌ي فايل‌ها در بسته و استخراج
                                #آن‌ها در مسير تعيين شده‌ي بالا
```

با کمک پارامتر **/oname** می‌توانید از مسیر پیش‌فرض نصب صرف نظر کنید و فایل مورد نظر را با نام و مسیر دلخواه خود ذخیره کنید. محدودیّت این پارامتر پشتیبانی نکردن از چندین فایل است.

```
SetOutPath "C:\AA"              #تعيين مسير خروجي فايل‌ها
File /oname="C:\BB\b.exe" "myApp\a.exe"   #صرف نظر از مسير پيش‌فرض و نام
                                          #فعلي فايل مبدأ
```

فراخوانی دستور **File** با نویسه‌های * و ? و همین‌گونه ارسال نام یک شاخه به دستور فقط فایل‌های موجود در شاخه را شامل می‌شود. برای قرار دادن همه‌ی فایل‌ها و همه‌ی شاخه‌های زیر مجموعه باید از پارامتر **/r** استفاده کنید.

```
File /r "myApp"                 #اين دستور شاخه‌ي مورد نظر و همه‌ي شاخه‌ها
                                #و فايل‌هاي زيرمجموعه را شامل مي‌شود
```

```
File /r "myApp\*.txt"           #فايل‌هاي متني موجود در شاخه و زيرمجموعه‌ها
```

دیگر پارامترهای قابل قبول برای این دستور عبارتند از:

- **/x** برای قرار ندادن گروهی از فایل‌ها در بسته. (برای اطّلاعات بیشتر مراجعه کنید به پیوست ب-۱)
- **/nonfatal** برای عدم ایجاد خطای جدی در صورت عدم وجود چیزی برای فشرده‌سازی
- **/a** برای نگهداری مشخّصات فایل‌ها در بسته و بازسازی آن‌ها هنگام استخراج

CopyFiles-۳-۹-۲

ممکن است بعضی از فایل‌های نصب را در کنار فایل اجرایی نصب قرار داده باشید. برای کپی این گونه فایل‌ها به مسیرهای دلخواه این دستور ما را یاری می‌کند.

```
CopyFiles "$EXEDIR\Files\*.*" "$INSTDIR"  #كپي همه‌ي فايل‌ها به شاخه‌ي مقصد
```

استفاده از این دستور به شکل ساده‌ی بالا ممکن است باعث نمایش روند کپی فایل‌ها در پنجره‌ی کپی ویندوز شود. برای جلوگیری از این کار باید پارامتر **/SILENT** را به این دستور ارسال کنید. همچنین با استفاده از پارامتر **/FILESONLY** باعث کپی شدن فقط فایل‌ها خواهید شد. و در نهایت قرار دادن یک عدد در انتهای دستور که معادل حجم فایل‌های کپی است، به **NSIS** در بهینه‌سازی و بررسی فضای خالی در شاخه‌ی مقصد کمک خواهید کرد.

```
CopyFiles "$EXEDIR\VBAME.dll" "$SYSDIR"          #نصب فايل مربوط به زبان‌هاي
                                                 #راست به چپ در شاخه‌ي سيستم
                                                 #(مربوط به ويژوال بيسيك)
```

```
CopyFiles /SILENT "$SYSDIR\kbdfa.dll" "$SYSDIR\dllcache"
```

تعیین مسیر خروجی به صورت کامل در این دستور الزامی است. در غیر این صورت نتایج غیر قابل پیش‌بینی خواهد بود.

MessageBox-۴-۹-۲

مورد استفاده‌ی این دستور نمایش پیام، پرسش، یا اخطار است. نخستین پارامتر این دستور گونه‌ی آیکون و دکمه‌ها ، و پارامتر دوّم متن پیام را تعیین می‌کنند.

```
MessageBox MB_OK "Hello"
```

در ادامه فهرست انواع مختلف پیام و دکمه‌های ممکن را مشاهده می‌کنید.

- MB_OK نمایش فقط یک دکمه‌ی OK
- MB_OKCANCEL نمایش دکمه‌های OK و Cancel
- MB_ABORTRETRYIGNORE دکمه‌های Abort، Retry، و Ignore
- MB_RETRYCANCEL دکمه‌های Retry و Cancel
- MB_YESNO دکمه‌های Yes و No
- MB_YESNOCANCEL دکمه‌های Yes، No، و Cancel
- MB_ICONEXCLAMATION نمایش آیکون اخطار (علامت تعجّب)
- MB_ICONINFORMATION نمایش آیکون اطّلاعات
- MB_ICONQUESTION نمایش آیکون پرسش
- MB_ICONSTOP نمایش آیکون هشدار
- MB_USERICON نمایش آیکون برنامه‌ی نصب
- MB_TOPMOST نمایش پیام جلوتر از همه‌ی پنجره‌ها
- MB_SETFOREGROUND اصرار بر انتقال پنجره‌ی پیام به جلو
- MB_RIGHT نمایش متن پیام در سمت راست پنجره
- MB_RTLREADING مورد استفاده برای پیام‌هایی که مانند فارسی با زبان راست به چپ هستند
- MB_DEFBUTTON1 به صورت پیش‌فرض نخستین دکمه انتخاب باشد
- MB_DEFBUTTON2 دوّمین دکمه انتخاب باشد
- MB_DEFBUTTON3 سوّمین دکمه انتخاب باشد

برای ترکیب کردن پارامترهای فوق باید از نویسه‌ی پایپ «|» استفاده کنیم

```
MessageBox MB_YESNO|MB_ICONQUESTION "Install?"  #آیکون پرسش با دکمه های#
                                                #Yes و No
```

پیامی که به این صورت در برنامه گنجانده شود حتّی در صورت ساکت بودن نصب هم به نمایش در خواهد آمد. برای جلوگیری از رخداد این اتّفاق از پارامتر SD/ به همراه شناسه‌ی دکمه‌ی پیش‌فرض استفاده می‌کنیم.

```
MessageBox MB_YESNO|MB_ICONINFORMATION "Install?" /SD IDYES
```

فهرست شناسه‌ی دکمه‌ها را در زیر مشاهده می‌کنید.

- IDABORT دکمه‌ی Abort
- IDCANCEL دکمه‌ی Cancel
- IDIGNORE دکمه‌ی Ignore
- IDNO دکمه‌ی No
- IDOK دکمه‌ی OK
- IDRETRY دکمه‌ی Retry
- IDYES دکمه‌ی Yes

برای بررسی اینکه پاسخ کاربر به پرسش چه بوده است، با کمک گرفتن از شناسه‌های فوق و تعیین یک برچسب، محلّی که باید به آن پرش شود را تعیین می‌کنیم.

```
Function .onInit  #اين تابع پيش از شروع نصب فراخواني مي‌شود

      MessageBox MB_YESNO|MB_ICONINFORMATION "Install?" IDYES ContinueInstall
      Abort       #درصورتي جواب منفي، نصب را ادامه نده

ContinueInstall:  #خروج از تابع و ادامه‌ي نصب
FunctionEnd
```

CreateShortCut-۵-۹-۲

ایجاد میانبر (**Shortcut**) یکی از نیازهای اصلی هنگام نصب یک برنامه است. این دستور ما را در انجام این کار یاری می‌کند. پارامترهای اجباری برای این دستور عبارتند از مسیر ایجاد میانبر و مسیر فایلی که میانبر به آن اشاره می‌کند.

```
CreateShortCut "$STARTMENU\Programs\myApp.lnk" "$INSTDIR\myApp.exe"
```

همان‌گونه که می‌دانید عبارات **$STARTMENU** و **$INSTDIR** در هنگام اجرای نصب با مسیر معادل خود در سیستم مقصد جایگزین می‌شوند. بنابراین ممکن است میان رشته‌های فوق نویسه‌ی فاصله قرار بگیرد. برای جلوگیری از چند قطعه شدن رشته‌های فوق توسّط نویسه‌ی فاصله حتما از گیومه یا آپستروف برای در برگرفتن پارامترها استفاده کنید.

دیگر پارامترهای قابل قبول برای این دستور به شرح زیر است. چنانچه تمایل به استفاده از پارامترهای آخر دارید باید پارامترهای ماقبل آن را با مقادیر مناسب پر کنید تا **NSIS** بتواند هدف شما را از ارسال پارامترها تشخیص دهد.

- پارامترهای ارسالی به برنامه‌ی هدف.
- نام و مسیر فایل آیکون میانبر ساخته‌شده.
- شماره‌ی ترتیبی آیکون در فایل حاوی آیکون.
- تنظیمات شروع.
- کلید میانبر برای اجرای فایل میانبر ایجاد شده.
- توضیحات فایل میانبر.

Sleep-۶-۹-۲

این دستور برای نمایش پیام، هشدار، یا پرسش از کاربر به کار می‌رود. این پیام می‌تواند پرسش برای نصب خودکار یک نرم‌افزار، پیام پایان یافتن عملیّات نصب یا هشدار برای انجام یک عملیّات غیر قابل بازگشت باشد.

ساختار فراخوانی این دستور به شکل زیر است.

```
Sleep 2000        #توقّف اجراي نصب به مدّت ۲۰۰۰ ميلي‌ثانيه يا ۲ ثانيه
```

CreateDirectory-۷-۹-۲

یکی از نیازهای اساسی برای نصب بسیاری از برنامه‌ها ایجاد شاخه است. این دستور شما را در انجام این کار یاری خواهد کرد. همان‌گونه که در دستور **File** مشاهده کردید می‌توان بدون استفاده از این دستور نیز شاخه‌های مورد نیاز را ایجاد نمود.

```
CreateDirectory "$PROGRAMFILES\AMIB Tools"
```

تعیین مسیر دقیق ایجاد شاخه برای این دستور الزامی است. بنابراین نمی‌توانید از مسیردهی مرتبط استفاده نمایید.

RMDir-۸-۹-۲

عملکرد این دستور در پاک کردن شاخه‌هاست. برای اینکه این دستور موفّق به حذف شاخه‌ی مورد نظر شود، غیر جاری بودن شاخه‌ی مورد نظر و خالی بودن آن الزامی است. برای حذف همه‌ی شاخه‌ها و فایل‌های زیر مجموعه باید از پارامتر **/r** استفاده کنیم.

```
SetOutPath "$SYSDIR"          #تغيير مسير جاري براي موفّق بودن حذف
RMDir /r "$INSTDIR"                  #حذف همه‌ي شاخه‌ها و فايل‌هاي زير مجموعه
```

چنانچه حذف بعضی از فایل‌ها هنگام نصب یا حذف برنامه مقدور نباشد، با ارسال پارامتر **/REBOOTOK** می‌توان پس از راه‌اندازی مجدّد ویندوز این کار را انجام داد. در چنین حالتی فلگ **Reboot** روشن خواهد شد و شما می‌توانید در صورت تمایل پس از پایان نصب سیستم را مجدّد راه‌اندازی کنید.

```
RMDir /r /REBOOTOK "$INSTDIR"

IfRebootFlag 0 +3       #پرش به سه دستور جلوتر در صورت عدم نياز
                        #به راه اندازي مجدّد و ادامه در صورت نياز

MessageBox MB_YESNO "Reboot now?" IDNO +2       #پرسش براي راه اندازي مجدّد
Reboot                  #راه اندازي مجدد سيستم
```

Rename-۹-۹-۲

برای تغییر نام فایل یا شاخه از این دستور استفاده می‌شود. علاوه بر تغییر نام می‌توانید از این دستور برای جا به جا کردن فایل‌ها و شاخه‌ها نیز کمک بگیرید. در هنگام تغییر نام و محل باید توجّه کنید که تغییر نام به یک فایل یا شاخه‌ی قبلاً موجود، امکان‌پذیر نیست و همین گونه تغییر محلّ شاخه‌ها به درایوی غیر از درایو مبدأ شکست خواهد خورد.

```
Rename "$PLUGINSDIR\myFile.dat" "$INSTDIR\myFile.exe"  #تغيير نام و محل
```

چنانچه مسیر دقیق را به این دستور ارسال نکنید مسیر فعلی که با دستور **SetOutPath** تعیین می‌شود مورد استفاده قرار خواهد گرفت و چنانچه مسیر فعلی را تعیین نکرده باشید، مسیر فعلی برابر با مسیر فایل نصب(**$EXEDIR**) خواهد بود.

مانند چندی دیگر از دستورات با کمک پارامتر **/REBOOTOK**، تغییر نام ناموفّق بعد از راه‌اندازی مجدّد سیستم انجام می‌شود (برای مشاهده‌ی مثال مراجعه کنید به دستور **RMDir**).

Delete-۱۰-۹-۲

حذف فایل وظیفه‌ی این دستور است. درست همانند دستورات **RMDir** و **Rename**، این دستور نیز از پارامتر **/REBOOTOK** پشتیبانی می‌کند.

برای حذف یک فایل تعیین مسیر و نام دقیق آن الزامی است. برای تعیین نام فایل می‌توانید از نویسه‌های * و ? استفاده کنید تا گروهی از فایل‌ها را هدف قرار دهید.

چنانچه **NSIS** قادر به حذف فایلی نباشد، پرچم خطا (**Error Flag**) روشن خواهد شد. ولی این اتّفاق تنها زمانی رخ می‌دهد که فایل موجود باشد.

```
Delete /REBOOTOK "$SYSDIR\myDll.dll"      #حذف يك فايل و تلاش مجدّد در صورت
                                          #ناموفّق بودن پس از شروع ويندوز
```

```
Delete "$INSTDIR\*.tmp"        #حذف گروهي از فايل‌ها با پسوند مشخّص
```

۲-۹-۱۱-Quit

برای خروج فوری از نصب این دستور را فراخوانی می‌کنیم. این دستور می‌تواند در نصب‌های بی‌صدا یا معمولی مورد استفاده قرار گیرد.

چنانچه از این دستور استفاده نکنید، برای بسته شدن نصب کاربر باید بر روی دکمه‌ی **Close** کلیک کند.

```
MessageBox MB_ICONINFORMATION "Installation completed successfully."
Quit
```

۲-۹-۱۲-دستورات رجیستری

ایجاد تغییرات در رجیستری ویندوز بنابر نیازهای مختلف کمابیش در هر نصبی لازم است.

در هنگام استفاده از دستورات رجیستری شاخه‌ی اصلی می‌تواند یکی از موارد زیر باشد.

- **HKCR** یا **HKEY_CLASSES_ROOT**
- **HKLM** یا **HKEY_LOCAL_MACHINE**
- **HKCU** یا **HKEY_CURRENT_USER**
- **HKU** یا **HKEY_USERS**
- **HKCC** یا **HKEY_CURRENT_CONFIG**
- **HKDD** یا **HKEY_DYN_DATA**
- **HKPD** یا **HKEY_PERFORMANCE_DATA**

رجیستری ویندوز تشکیل شده از شاخه و متغیّر که متغیّرها می‌توانند گونه‌های مختلفی داشته باشند. برای ایجاد، پاک‌کردن، و پردازش شاخه‌ها و متغیّرها دستورات زیر در اختیار شما هستند:

- **DeleteRegKey** – پاک‌کردن شاخه‌ها از رجیستری با این دستور امکان‌پذیر است. پارامترهای این دستور عبارتند از نام شاخه‌ی اصلی و شاخه‌ی فرعی برای حذف. استفاده از پارامتر **/ifempty** باعث تنها پاک شدن شاخه‌های خالی می‌شود.

```
DeleteRegKey HKLM "Software\AMIB"                #حذف شاخه و تمام زير شاخه‌ها

DeleteRegKey /ifempty HKLM "Software\AMIB"       #حذف شاخه در صورت خالي بودن
```

- **DeleteRegValue** – با کمک این دستور می‌توان متغیّر دلخواه را از رجیستری پاک کرد. در صورت عدم وجود متغیّر در رجیستری یا رخداد اشکال در فرآیند، پرچم خطا روشن خواهد شد.

```
DeleteRegValue HKLM "Software\AMIB" "AppPath"
```

- **EnumRegKey** – برای به دست آوردن فهرست شاخه‌های موجود در رجیستری، از این دستور استفاده می‌شود. علاوه بر پارامترهای تعیین کننده‌ی مسیر، یک پارامتر عددی که تعیین کننده‌ی شماره‌ی ترتیبی شاخه‌ی مورد نظر است، نیز باید به دستور **EnumRegKey** ارسال شود. چنانچه شماره‌ی ارسال شده فراتر از محدوده‌ی تعداد شاخه‌ها باشد، مقدار برگشتی خالی خواهد بود. و چنانچه خطایی رخ دهد، پرچم خطا نیز روشن خواهد شد.

```
StrCpy $0 0                          #قرار دادن مقدار صفر در متغیّر
loop:                                #تعیین برچسب براي شروع حلقه
  EnumRegKey $1 HKLM Software $0     #قرار دادن نام شاخه‌ي موردنظر در متغیّر
  StrCmp $1 "" done                  #پرش به پایان حلقه در صورت نبودن شاخه
  IntOp $0 $0 + 1                    #اضافه‌ي یك واحد به متغیّر عددي
  #نمایش نام شاخه‌ي خوانده شده از رجیستري
  MessageBox MB_YESNO|MB_ICONQUESTION "$1$\n$\nMore?" IDYES loop
done:                                #تعیین برچسب براي پایان حلقه
```

- **EnumRegValue** – برای ایجاد فهرست متغیّرهای موجود در یک شاخه از این دستور استفاده می‌کنیم. روش فراخوانی درست همانند دستور **EnumRegKey** است. با این تفاوت که در برابر برگشت دادن شاخه‌ها، متغیّرها فهرست خواهند شد.

```
StrCpy $0 0          #مقداردهي متغیّر با عدد صفر
loop:                #تعیین برچسب براي شروع حلقه
ClearErrors          #پاك كردن پرچم خطا براي تشخیص دادن خطا

#خواندن ترتیبي متغیّرهاي رجیستري با توجه به عدد ارسالي
EnumRegValue $1 HKLM Software\Microsoft\Windows\CurrentVersion $0

IfErrors done        #پرش به پایان حلقه در صورت رخداد خطا
IntOp $0 $0 + 1      #اضافه كردن یك واحد به متغیّر ترتیبي

#خواندن مقدار متغیّر فهرست شده
ReadRegStr $2 HKLM Software\Microsoft\Windows\CurrentVersion $1

#نمایش نام متغیّر و مقدار آن و پرسش براي ادامه‌ي كار
MessageBox MB_YESNO|MB_ICONQUESTION "$1 = $2$\n$\nMore?" IDYES loop
done:
```

- **ReadRegDWORD** – همان‌گونه که گفته‌شد، متغیّرهای رجیستری انواع مختلفی را می‌پذیرند. برای خواندن مقدار عددی (۳۲ بیتی) این دستور را فراخوانی می‌کنیم. متغیّر خروجی، شاخه‌ی اصلی، شاخه‌ی فرعی، و نام متغیّر رجیستری پارامترهای این دستور هستند. در صورت نبودن متغیّر مورد نظر، مقدار خالی برگردانده می‌شود و پرچم خطا روشن خواهد شد. و چنانچه متغیّر مورد نظر عددی نباشد، مقدار آن خوانده می‌شود ولی پرچم خطا نیز روشن می‌گردد.

```
ReadRegDWORD $0 HKLM "Software\NSIS" "VersionBuild"   #NSIS خواندن نگارش
```

- **ReadRegStr** – خواندن مقدار متغیّرهای رشته‌ای وظیفه‌ی این دستور است. در صورت نبود متغیّر، مقدار برگشتی خالی خواهد بود و پرچم خطا نیز روشن می‌شود. عددی بودن متغیّرِ اشاره‌شده باعث خوانده شدن عدد به صورت رشته‌ای و نیز ایجاد یک خطا می‌گردد.

```
ReadRegStr $0 HKLM Software\NSIS ""        #خواندن مقدار متغیّر پیشفرض
DetailPrint "NSIS is installed at: $0"     #نمایش مقدار خوانده‌شده در لیست
```

- **WriteRegBin** – برای نوشتن متغیّرهای باینری در رجیستری این دستور را فراخوانی می‌کنیم. مقدار ارسالی باید در مبنای ۱۶ باشد. عدم وجود شاخه‌ی اشاره‌شده باعث ایجاد آن می‌شود و چنانچه خطایی در هنگام نوشتن رخ دهد، پرچم خطا روشن می‌شود.

```
WriteRegBin HKLM "Software\My Company\My Software" "Binary Value" DEADBEEF01251
```

- **WriteRegDWORD** – نوشتن مقادیر عددی ۳۲ بیت در رجیستری با این دستور امکان‌پذیر است.

```
WriteRegDWORD HKLM "Software\My Company\My Software" "DWORD Value 1" 0xA2F4

WriteRegDWORD HKLM "Software\My Company\My Software" "DWORD Value 2" 1234
```

- **WriteRegStr** – ایجاد یا ویرایش متغیّرهای رجیستری با گونه‌ی **REG_SZ** با این دستور انجام می‌پذیرد.

```
WriteRegDWORD HKLM "Software\My Company\My Software" "Name" "AMIB"
```

- **WriteRegExpandStr** – ایجاد یا ویرایش متغیّرهای رجیستری با گونه‌ی **REG_EXPAND_STR** با کمک این دستور امکان‌پذیر است. مورد استفاده‌ی این گونه متغیّر در رشته‌هایی است که خود حاوی متغیّر هستند. مانند **%windir%** که باید با شاخه‌ی ویندوز جایگزین شود.

```
WriteRegDWORD HKLM "Software\My Software" "AppPath" "%WINDIR%\AMIB"
```

۲-۹-۱۳-دستورات فایل ini

اگرچه فایل‌های **ini** در نرم‌افزارهای امروزی جایگاهی ندارند ولی گاهی اوقات ایجاد تغییرات در این گونه فایل‌ها نیاز است. برای ایجاد تغییرات در این فایل‌ها می‌توانید از دستورات خواندن و نوشتن فایل نیز استفاده کنید ولی پردازش بخش‌ها و متغیّرهای موجود در آن بر عهده‌ی شما خواهد بود.

در بخش‌های زیر فهرست دستورات قابل قبول و شرح کوتاهی از عملکرد آن‌ها را مشاهده خواهید کرد.

- **DeleteINISec** – حذف کامل یک بخش از فایل. این دستور دو پارامترِ نام فایل و نام بخش را می‌پذیرد و چنانچه قادر به حذف بخش نباشد، پرچم خطا را روشن خواهد کرد.
- **DeleteINIStr** – حذف یک متغیّر از یک بخش وظیفه‌ی این دستور است. پارامترهای این دستور عبارتند از نام فایل، نام بخش و نام متغیّری که باید حذف شود.
- **FlushINI** – برای اعمال تغییرات انجام شده بر روی فایل اصلی از این دستور استفاده می‌کنیم. ویندوز ممکن است تغییراتی را که انجام می‌دهیم در حافظه نگهداری کند و آن‌ها را به دیسک سخت انتقال ندهد (برای افزایش کارایی). بنابراین پس از اجرای همه‌ی دستورات بر روی فایل مورد نظر، از این دستور استفاده می‌کنیم.
- **ReadINIStr** – خواندن مقدار حرفی یک متغیّر وظیفه‌ی این دستور است. پارامترهای لازم برای این دستور عبارتند از یک متغیّر برای نگهداری مقدار خروجی، نام فایل، نام بخش و در پایان نام متغیّری که مقدار آن باید خوانده شود.
- **WriteINIStr** – برای نوشتن یک متغیّر در یک بخش از این دستور استفاده می‌کنیم. این دستور نیازمند پارامترهای نام فایل، نام بخش، نام متغیّر، و مقدار متغیّر است.

برای فراگیری بهتر شیوه‌ی استفاده از دستورات **INI** به مثال ذیل و توضیحات آن توجّه کنید:

```
SetOutPath $INSTDIR                 #تغيير مسير فعلي به مسير اصلي نصب برنامه
WriteINIStr "myFile.ini" "mySection" "myVar" "TRUE" #نوشتن متغيّر در فايل

ReadINIStr $0 "myFile.ini" "mySection" "myVar"  #خواندن مقدار متغيّر از فايل
MessageBox MB_ICONINFORMATION $0    #نمايش مقدار متغيّر خوانده شده

FlushINI                            #ذخيره‌ي تغييرات فايل بر روي ديسك سخت

Delete "$INSTDIR\myFile.ini"        #حذف فايل ايجاد شده از روي ديسك سخت
```

۲-۹-۱۴-دستورات اجرای فایل خارجی

برای اجرای فایل‌ها و فرامین در زمان نصب چند دستور با کارایی‌های مختلف وجود دارند که در ادامه به فهرست و شرح عملکرد آن‌ها خواهیم پرداخت.

- **Exec** – ساده‌ترین شکل اجرای برنامه‌های خارجی با استفاده از این دستور امکان‌پذیر است. تنها پارامتر این دستور نام فایلی است که باید اجرا شود. نکته‌ی قابل ذکر در استفاده‌ی این دستور لزوم به کاربردن گیومه (") در ابتدا و انتهای نام فایل اجرایی است. چنانچه از گیومه استفاده نکنید و نام یا مسیر فایل حاوی نویسه‌ی فاصله باشد، برنامه‌ی شما در ویندوزهای **9x** اجرا نخواهد شد. اجرای برنامه با کمک این دستور باعث بازگشت فوری روند اجرا به نصب خواهد شد.

```
Exec '"$INSTDIR\Setup2.exe" /SecretWord'          #اجراي برنامه با پارامتر
```

- **ExecShell** – فراخوانی برنامه‌ی مربوط به انواع غیر اجرایی از فایل‌ها وظیفه‌ی این دستور است. برای مثال باز کردن یک فایل متنی در ویرایشگر پیش‌فرض متن. این دستور چند پارامتر می‌پذیرد که عبارتند از نام دستوری که باید بر روی فایل انجام شود، نام فایل، پارامترها، و شیوه‌ی نمایش پنجره‌ی جدید. برای روشن‌تر شدن شیوه‌ی استفاده به مثال و فهرست شیوه‌های نمایش که در ادامه آمده توجّه کنید.

```
ExecShell "" "$INSTDIR\ReadMe.txt"  #باز كردن فايل متني با دستور پيشفرض
```

– **SW_SHOWNORMAL** – حالت پیش‌فرض – نمایش پنجره به صورت عادّی.
– **SW_SHOWMAXIMIZED** – نمایش پنجره به در حالت بیشینه.
– **SW_SHOWMINIMIZED** – نمایش پنجره در حالت کمینه.
– **SW_HIDE** – پنجره‌ی مخفی.

```
#اجراي فرمان خاص براي باز كردن فايل متني و نمايش پنجره در حالت بيشينه
ExecShell "Open" "$INSTDIR\ReadMe.txt" "" SW_SHOWMAXIMIZED

ExecShell "Print" "$INSTDIR\ReadMe.txt"   #چاپ فايل متني با برنامه‌ي مرتبط

ExecShell "Open" "http://nsis.sf.net/"    #باز كردن سايت با مرورگر پيشفرض
```

- **ExecWait** – اجرای فایل و انتظار برای پایان یافتن عملکرد آن وظیفه‌ی این دستور است. همه‌ی نکات ذکر شده در دستور **Exec** در این دستور نیز قابل توجّه هستند. برای دریافت کد خروجی (**Exit Code**) از برنامه‌ی اجرا شده می‌توانید یک متغیّر را به عنوان پارامتر در انتهای دستور **ExecWait** قرار دهید.

```
ExecWait '"$INSTDIR\myProg.exe"' $0 #اجراي برنامه و دريافت خروجي در متغيّر
MessageBox MB_ICONINFORMATION "Exit Code was : $0"    #نمايش كد خروجي
```

همان گونه که مشاهده می‌کنید نام فایل اجرایی باید درون گیومه قرار بگیرد و آپستروف نیز برای تعیین رشته در **NSIS** به کار رفته است.

۲-۹-۱۵-دستورات خواندن، نوشتن، و جستجوی فایل

برای ایجاد تغییرات در فایل‌ها، ساخت فایل‌های جدید، و جستجوی فایل‌ها دستوراتی در اختیار شماست که در ادامه فهرست و توضیحات آن‌ها را مشاهده می‌کنید:

- **FileOpen** – پیش از انجام هر عملی بر روی فایل‌ها ابتدا باید آن‌ها را باز کنیم. برای این کار کافیست این دستور را با سه پارامتر متغیّر خروجی، نام فایل، و روش باز کردن فراخوانی کنیم. روش‌های باز کردن فایل عبارتند از **w** برای نوشتن،

r برای خواندن، و **a** برای اضافه کردن به انتهای فایل. فراموش نکنید که پس از پایان عملیّات خواندن و نوشتن حتماً فایل را با دستور **FileClose** ببندید. چنانچه مسیر قرارگیری فایل را تعیین نکنید مسیر فعلی که با دستور **SetOutPath** تعیین می‌شود مورد استفاده قرار خواهد گرفت.

- **FileClose** – برای بستن فایلی که قبلاً با دستور **FileOpen** باز شده است، استفاده می‌شود. تنها پارامتر این دستور دستگیره‌ی فایلی است که باز شده است.

```
FileOpen $0 "myFile.dat" r     #باز كردن فايل براي خواندن و ذخيره‌ي
                               #دستگيره‌ي آن در متغيّر ارسالي
FileClose $0                   #بستن فايل باز شده
```

- **FileRead** – برای خواندن رشته‌ها از داخل فایل این دستور ما را یاری خواهد کرد. در چند حالت خواندن رشته متوقّف می‌شود که عبارتند از رسیدن به نویسه‌ی کد صفر، رسیدن به پایان خط، رسیدن به محدودیّت خواندن کاربر، و رسیدن به محدودیّت ۱۰۲۴ بایتی رشته‌ها در **NSIS**. در صورت تمایل می‌توانید نسخه‌های ویرایش شده‌ی **NSIS** که محدودیّت ۱۰۲۴ بایتی آن‌ها بیشتر باشد را از سایت **nsis.sf.net** دانلود کنید. دستور **FileRead** سه پارامتر می‌پذیرد که عبارتند از دستگیره‌ی فایل باز شده، متغیّر خروجی برای ذخیره‌ی رشته، و در صورت تمایل مقداری عددی برای محدود کردن طول رشته.

```
FileOpen $0 "myFile.txt" r     #باز كردن فايل متني براي خواندن

FileRead $0 $1                 #خواندن رشته از ابتداي فايل و ذخيره در متغير
DetailPrint $1                 #نوشتن رشته‌ي خوانده شده در ليست عمليّات

FileRead $0 $1 100             #خواندن بيشينه ۱۰۰ بايت از ادامه‌ي فايل
DetailPrint $1                 #نوشتن رشته‌ي خوانده شده در ليست عمليّات

FileClose $0                   #بستن فايل باز شده
```

- **FileReadByte** – وظیفه‌ی این دستور خواندن یک بایت از فایل باز شده با دستور **FileOpen** است. بایت خوانده شده به صورت عددی در متغیّر ارسالی ذخیره می‌شود. چنانچه اشاره‌گر محلّ فایل به انتهای آن برسد و این دستور قادر به خواندن مقداری نباشد، مقدار بازگشتی خالی خواهد بود و پرچم خطا روشن می‌شود.

```
FileReadByte $0 $1      #خواندن يك بايت به صورت عددي از فايل باز شده
FileReadByte $0 $2      #خواندن بايت بعدي از فايل

DetailPrint "$1 $2"     #چاپ دو عدد خوانده شده در ليست عمليّات نصب
```

- **FileSeek** – برای تغییر دادن محلّ فعلی در فایل باز شده از این دستور استفاده می‌کنیم. پارامترهای این دستور عبارتند از دستگیره‌ی فایل، محلّ پرش، گونه‌ی پرش، و متغیّری برای بازگرداندن محلّ فعلی فایل. دو پارامتر آخر اختیاری هستند و می‌توانید از آن‌ها استفاده نکنید. حالت پیش‌فرض برای پارامتر «گونه‌ی پرش» **SET** است. در حالت **SET** محلّ جدید وابسته به ابتدای فایل و محلّ پرش تعیین می‌شود. برای پرش به صورت مرتبط از مقدار **CUR** استفاده می‌کنیم تا پرش متناسب با محلّ فعلی انجام شود و در نهایت برای پرش متناسب با نقطه‌ی انتهایی فایل از مقدار **END** استفاده می‌کنیم.

```
FileSeek $0 100          #پرش به بایت ۱۰۰ از ابتدای فایل
FileSeek $0 100 SET      #پرش به بایت ۱۰۰ از ابتدای فایل

FileSeek $0 +2 CUR       #پرش به دو بایت جلوتر متناسب با محلّ فعلی
FileSeek $0 -3 CUR       #پرش به سه بایت عقبتر متناسب با محلّ فعلی

FileSeek $0 -4 END       #پرش به چهار بایت قبل از انتهای فایل

FileSeek $0 +5 CUR $1    #پرش به پنج بایت جلوتر و ذخیره‌ی محل جدید در متغیّر
```

- **FileWrite** – برای نوشتن رشته‌ی حرفی در فایل باز شده، این دستور ما را یاری می‌دهد. دو پارامتر مورد نیاز برای اجرای این دستور دستگیره‌ی فایل و رشته‌ی مورد نظر هستند. چنانچه در حین نوشتن خطایی رخ دهد پرچم خطا روشن خواهد شد.

```
ClearErrors                          #حذف پرچم خطا برای جلوگیری از اشتباه
FileOpen $0 "$INSTDIR\File.txt" w    #باز کردن فایل جدید برای نوشتن
IfErrors Error                       #پرش به پایان در صورت رخداد خطا
FileWrite $0 "Hello NSIS.$\n"        #نوشتن رشته‌ی حرفی و رفتن به خط بعد
FileClose $0                         #بستن فایل باز شده

Error:
```

- **FileWriteByte** – برای نوشتن یک بایت در فایل باز شده، از این دستور استفاده می‌شود. دو پارامتر اجباری آن عبارتند از دستگیره‌ی فایل باز شده و رشته‌ی حاوی عدد بایت مورد نظر.

```
FileWriteByte $0 "13"                #نوشتن نویسه‌ی شماره‌ی ۱۳ در فایل
FileWriteByte $0 65                  #نوشتن نویسه‌ی شماره‌ی ۶۵ در فایل (A)
```

- **FindFirst** – برای جستجو کردن یک فایل یا گروهی از فایل‌ها از سه دستور **FindFirst**، **FindNext**، و **FindClose** استفاده می‌شود. جهت شروع جستجو و تعیین قالب جستجو **FindFirst** را با پارامترهای دستگیره‌ی جستجو (خروجی)، نخستین نتیجه (خروجی)، و قالب جستجو (ورودی) فراخوانی می‌کنیم. چنانچه فایلی با مشخّصات مورد نظر یافت نشود، دو متغیّر ارسالی مقدار خالی می‌گیرند و پرچم خطا روشن خواهد شد.

```
FindFirst $0 $1 "$SYSDIR\*.dll"      #جستجوی گروهی از فایل‌ها در شاخه‌ی
                                     #سیستم، و ذخیره‌ی دستگیره‌ی جستجو و نام
                                     #نخستین فایل در متغیّرهای ارسالی
```

با اجرای مثال بالا، **$0** حاوی دستگیره‌ی جستجو و **$1** نگهدارنده‌ی نام نخستین فایل پیدا شده با پسوند **dll.** خواهد بود.

- **FindNext** – پس از پیدا شدن نخستین فایل، برای جستجوی فایل‌های بعدی از **FindNext** استفاده می‌کنیم. این دستور دو پارامتر ورودیِ دستگیره‌ی جستجو و خروجیِ نام فایل را دریافت می‌کند.

```
FindNext $0 $1                       #جستوی فایل بعدی دارای شرایط مورد نظر
```

- **FindClose** – پس از پایان یافتن فرآیند جستجو و به دست آمدن نتایج مورد نظر، با فراخوانی این دستور جستجوی باز شده را می‌بندیم. در ادامه به یک مثال کامل از شیوه‌ی جستجوی گروهی از فایل‌ها توجّه کنید.

```
FindFirst $0 $1 "$SYSDIR\*.ocx"     #شروع جستجو براي فايل‌هاي كامپونت

loop:                               #تعيين برچسب براي ايجاد حلقه
      StrCmp $1 "" done             #پرش به پايان در صورت پيدا نشدن فايل
      DetailPrint $1                #چاپ نام فايل در ليست عمليّات
      FindNext $0 $1                #جستجوي دوباره براي يافتن فايل بعدي
      Goto loop                     #پرش به ابتداي حلقه
done:                               #تعيين برچسب براي پايان حلقه

FindClose $0                        #بستن جستجوي باز شده پس از پايان جستجو
```

۲-۱۰-حذف (Uninstall)

همان گونه که مشاهده کردید فایل اسکریپت **NSIS** از سه پهنه‌ی سراسری، توابع، و بخش‌ها تشکیل یافته است. دستورات مربوط به حذف برنامه نیز می‌تواند در میان همین دستورات قرار بگیرند. به غیر از توابع و بخش‌ها که بر اساس گونه‌ی تعریف فقط در بخش نصب یا حذف قابل دسترسی هستند، دستوراتی که در پهنه‌ی سراسری مورد استفاده قرار بگیرند هم حذف و هم نصب را تحت تأثیر قرار می‌دهند.

ترتیب استفاده از دستورات نصب و حذف تأثیری در خروجی ندارد و هر دستور در هر مکانی می‌تواند قرار بگیرد.

هنگامی که در مورد صفحات گفته شد، علاوه بر دستور **Page** که وظیفه‌ی تعیین ترتیب صفحات نصب را بر عهده داشت، با دستور **UninstPage** نیز آشنا شدیم که درست همانند دستور **Page** عمل می‌کند و ترتیب نمایش صفحات حذف را مشخّص می‌نماید.

در مورد توابع نیز آموختیم که برای ایجاد توابع زمان حذف قرار دادن پیشوند **un.** برای این گونه توابع الزامی است و فراخوانی توابع زمان نصب توسّط بخش‌های حذف امکان‌پذیر نیست و همین‌گونه حالت عکس.

هنگامی که برنامه‌ی حذف فراخوانی شود، بخش‌هایی که پیشوند **un.** داشته باشند یا با نام **Uninstall** تعریف شده باشند، به ترتیب قرارگیری در فایل اسکریپت اجرا خواهند شد.

ایجاد بخش‌ها و توابع حذف به تنهایی باعث ایجاد فایل اجرایی حذف برنامه نمی‌شوند. برای انجام این کار باید در هنگام نصب دستور **WriteUninstaller** را فراخوانی کنید. این دستور یک پارامتر که مسیر و نام برای ذخیره‌ی فایل اجرایی حذف است، می‌پذیرد. فراخوانی این دستور فقط در بخش‌ها و توابع نصب امکان‌پذیر است و در صورتی عمل می‌کند که دست‌کم یک بخش با شرایط یاد شده برای حذف موجود باشد. برای ایجاد چند فایل اجرایی حذف می‌توانید این دستور را چند بار صدا بزنید.

```
WriteUninstaller "$INSTDIR\UnInstall.exe" #ايجاد فايل اجرايي حذف برنامه
```

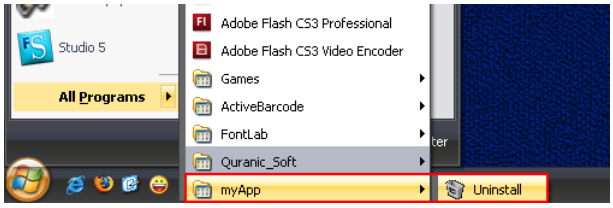
فراموش نکنید که نخستین گام در هنگام حذف برنامه، پاک کردن فایل اجرایی حذف است. در سیستم‌عامل ویندوز پاک کردن فایل اجراییِ در حال اجرا، ممکن نیست. برای رفع این مشکل، هنگامی که فایل حذف برنامه را اجرا می‌کنید، **NSIS** یک کپی از این فایل را در شاخه‌ی **TEMP** کاربر ایجاد می‌کند و آن را اجرا می‌کند. بنابراین بدون ایجاد مشکل می‌توانید فایل اجرایی حذف را با دستور **Delete** پاک کنید.

```
Section "Uninstall"                 #شروع بخش حذف
Delete "$INSTDIR\Uninstall.exe"     #پاک کردن فایل اجرایی حذف بدون مشکل
SectionEnd                          #پایان بخش حذف
```

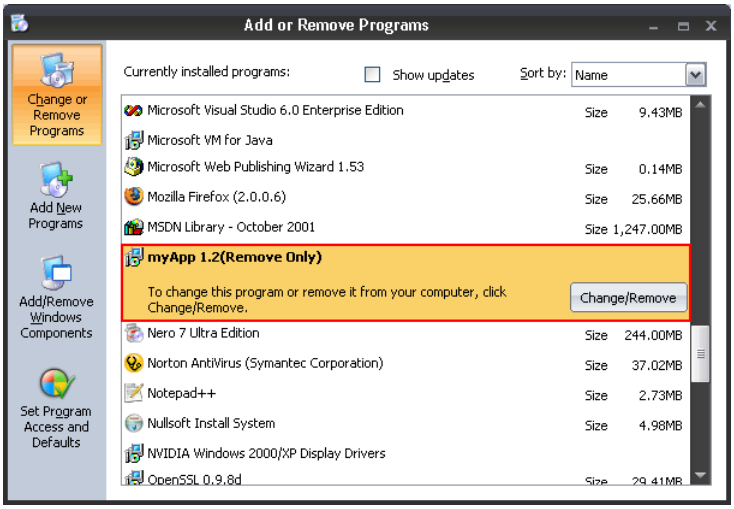
حذف برنامه به درخواست کاربر انجام می‌شود، بنابراین کاربر باید راهی ساده برای اجرای برنامه‌ی حذف در اختیار داشته باشد. می‌توانید با فراخوانی دستور **CreateShortcut** و ایجاد یک فایل میانبر برای برنامه‌ی حذف، این کار را انجام دهید و همین گونه با ایجاد بخش‌های مربوط در رجیستری ویندوز، امکان حذف برنامه از راه **Control Panel** ویندوز را برای کاربر فراهم نمایید.

```
OutFile "UnInst Test.exe"                   #نام فایل خروجی نصب - اجباری
InstallDir "$PROGRAMFILES\UninstTest"       #تعیین مسیر پیشفرض نصب برنامه

Page directory                              #تعیین ترتیب صفحات نصب
Page instfiles                              #

UninstPage uninstconfirm                    #تعیین ترتیب صفحات حذف
UninstPage instfiles                        #

Name "UnInst Test"                          #نام بسته

Section "Install"                           #بخش مربوط به نصب نرم افزار

SetOutPath "$INSTDIR"                       #ایجاد شاخه‌ی برنامه
WriteUninstaller "$INSTDIR\Uninst.exe"      #ایجاد فایل اجرایی حذف برنامه
# Start Menu ایجاد شاخه‌ی برنامه در
CreateDirectory   "$STARTMENU\Programs\myApp"

#ایجاد فایل میانبر برای حذف برنامه
CreateShortcut    "$STARTMENU\PROGRAMS\myApp\Uninstall.lnk" \
                  "$INSTDIR\Uninst.exe"

#نوشتن نام برنامه در رجیستری برای حذف
WriteRegStr HKLM  "Software\Microsoft\Windows\CurrentVersion\Uninstall\myApp" \
                  "DisplayName" "myApp 1.2(Remove Only)"

#نوشتن فرمان لازم برای حذف برنامه در رجیستری
WriteRegStr HKLM  "Software\Microsoft\Windows\CurrentVersion\Uninstall\myApp" \
                  "UninstallString" "$INSTDIR\Uninst.exe"

SectionEnd                                  #پایان بخش نصب

Section "Uninstall"                         #بخش حذف برنامه

#پاک کردن فایل اجرایی حذف از دیسک سخت
Delete "$INSTDIR\UnInst.exe"
#پاک کردن فایل میانبر حذف
Delete "$STARTMENU\Programs\myApp\Uninstall.lnk"
# Start Menu حذف شاخه‌ی برنامه در
Rmdir "$STARTMENU\Programs\myApp"
#پاک کردن شاخه‌ی حذف برنامه در رجیستری ویندوز
DeleteRegKey HKLM "Software\Microsoft\Windows\CurrentVersion\Uninstall\myApp"

SectionEnd                                  #پایان بخش حذف
```

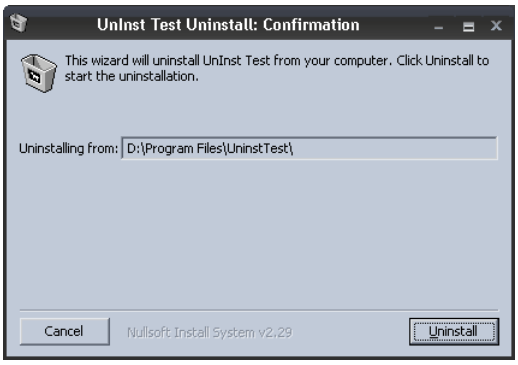
- تصویر فایل میانبر حذف ایجاد شده در **Start Menu**



- تصویر عنوان ایجاد شده در **Add or Remove Programs**



- تصویر پنجره‌ی حذف پس از اجرای فایل میانبر یا عنوان موجود در **Control Panel**



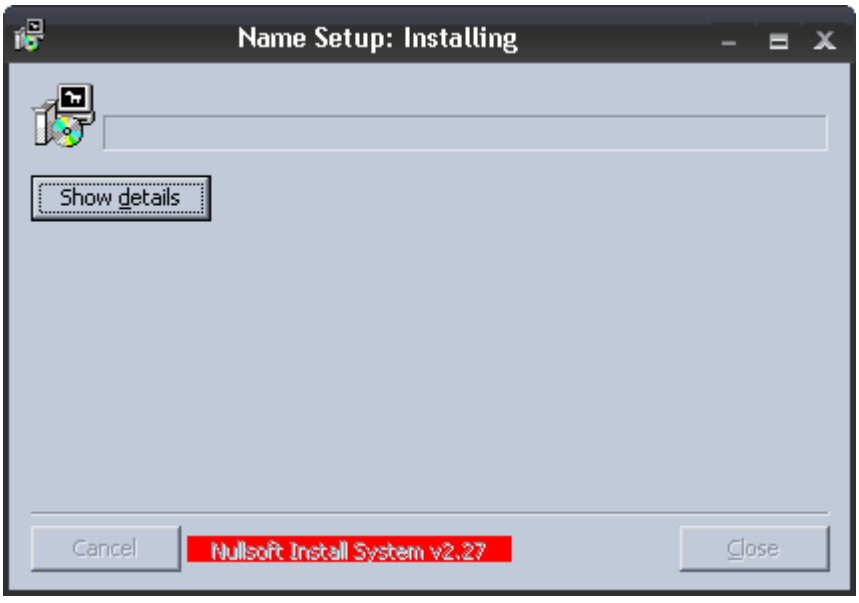
پیوست الف: فهرست توابع CallBack

الف-۱- توابع CallBack در زمان نصب

الف-۱-۱- .onGUIInit

این تابع زمانی فراخوانی می‌شود که همه‌ی منابع لازم برای اجرای نصّاب بارگزاری شده‌اند و گام بعدی نمایش پنجره‌ی نصب است. استفاده‌ی این تابع می‌تواند ایجاد تغییرات در ظاهر برنامه باشد.

```
!include "WinMessages.nsh"

Function .onGUIInit
      # ۱۰۲۸ کد مربوط به شيء معرّفي نصّاب است
      GetDlgItem $R0 $HWNDPARENT 1028
      # ايجاد فونت جديد در حافظه
      CreateFont $R1 "Tahoma" 8 500
      # تغيير فونت شيء مورد نظر
      SendMessage $R0 ${WM_SETFONT} $R1 0
      # تغيير رنگ زمينه به سفيد و رنگ متن به قرمز
      SetCtlColors $R0 FFFFFF FF0000
FunctionEnd
```



الف-۱-۲- .onInit

این تابع زمانی فراخوانی می‌شود که تقریباً همه‌ی مراحل لازم برای شروع نصب به پایان رسیده است. با استفاده از این تابع و اجرای دستور **Abort** در آن می‌توان از اجرای نصب جلوگیری کرد.

```
Function .onInit
      ; خواندن يك رشته از داخل فايل
      ReadINIStr $INSTDIR $WINDIR\wincmd.ini Configuration InstallDir
      StrCmp $INSTDIR "" 0 NoAbort
      MessageBox MB_OK "برنامه‌ي مورد نظر روي سيستم شما يافت نشد"
      Abort       ; خروج از نصّاب
      NoAbort:    ; ليبل براي پرش و ادامه‌ي نصب
FunctionEnd
```

با استفاده از این تابع می‌توان موارد لازم برای اجرای نصب را بررسی کرد و در صورت لزوم از برنامه‌ی نصب خارج شد.

الف-۱-۳- onInstFailed.

زمان فراخوانی این تابع هنگامی است که کاربر پس از ایجاد یک خطای غیر قابل اصلاح بر روی **Cancel** کلیک کند.

الف-۱-۴- onInstSuccess.

پس از پایان موفّقیّت‌آمیر نصب این تابع فراخوانی می‌شود. موارد استفاده‌ی آن می‌تواند پرسش برای اجرای برنامه‌ی نصب شده، نمایش راهنمای برنامه باشد و یا سپاسگزاری از کاربر برای نصب برنامه.

الف-۱-۵- onGUIEnd .

اجرای این تابع پس از بسته‌شدن پنجره‌ی نصب است و می‌تواند برای آزادسازی منابع اشغال شده توسط کاربر استفاده شود.

الف-۱-۶- onMouseOverSection.

هنگامی که کاربر نشانگر ماوس رو بر روی بخش‌های نصب وارد کند، این تابع فراخوانی می‌شود. مور استفاده‌ی این تابع عوض کردن راهنمای بخش است. برای مشاهده‌ی روش انجام این کار به مثال‌های موجود در **NSIS** مراجعه کنید.

الف-۱-۷- onRebootFailed.

همان‌گونه که از نام این تابع آشکار است، هنگامی که به هر دلیل نیاز به راه‌اندازی مجدّد سیستم باشد و **NSIS** نتواند این کار را انجام دهد این تابع را فراخوانی می‌کند.

الف-۱-۸- onSelChange.

زمانی که کاربر تغییراتی در انتخاب بخش‌های نصب ایجاد کند، یا گونه‌ی نصب را تغییر دهد این تابع صدا زده می‌شود.

الف-۱-۹- onUserAbort.

هنگامی که کاربر نصب را بدون اینکه مشکلی پیش از آن رخ داده باشد، لغو کند، این تابع فراخوانی می‌شود.

الف-۱-۱۰- onVerifyInstDir.

در هنگام نصب کاربر می‌تواند شاخه‌ی نصب را تغییر دهد. با فراخوانی این تابع در این مواقع می‌توانید اجازه‌ی نصب در شاخه‌ی مورد نظر را به وی بدهید یا از نصب جلوگیری نمایید.

برای جلوگیری از نصب برنامه در شاخه‌ی انتخاب شده کافیست از دستور **Abort** در تابع ذکر شده استفاده کنید:

```
Function .onVerifyInstDir
      IfFileExists "$INSTDIR\Winamp.exe" PathGood
      Abort ; چنانچه شاخه‌ي مقصد شاخه‌ي وينمپ نبود، اجازه‌ي نصب نده
      PathGood:
FunctionEnd
```

الف-۲- توابع CallBack در زمان UnInstall

الف-۲-۱- un.onGUIInit

در این تابع می‌توان دستورات لازم برای ایجاد تغییرات در ظاهر برنامه را گنجاند. به تابع مشابه در زمان نصب مراجعه کنید.

الف-۲-۲- un.onInit

فراخوانی این تابع پس از بارگزاری منابع مورد نیاز برای **UnInstaller** صورت می‌پذیرد. در این گام می‌توان حذف برنامه را لغو کرد یا پرسش برای لغو را در این گام قرار داد. همچنین می‌توان شاخه‌ی مقصد که برنامه در آن نصب شده را نیز در این گام بررسی و اصلاح کرد.

الف-۲-۳-un.onUninstFailed

تابع مورد نظر هنگامی صدا زده می‌شود که اشکالی در عملیّات رخ دهد و کاربر خروج از برنامه را انتخاب کند.

الف-۲-۴-un.onUninstSuccess

زمانی که برنامه به درستی از روی سیستم حذف شود و پیش از خروج این تابع فراخوانی می‌شود.

الف-۲-۵-un.onGUIEnd

فراخوانی پس از بسته شدن پنجره‌ی **NSIS** برای انجام کارهای لازم جهت پاکسازی حافظه و حذف منابع استفاده شده.

الف-۲-۶-un.onRebootFailed

فراخوانی این تابع نشان می‌دهد که راه‌اندازی مجدّد سیستم قابل انجام نبوده است. محدودیّت‌هایی در اجرای دستورات رجیستری، ایجاد فایل **UnInstaller**، پردازش فایل‌ها، و استفاده از پلاگین‌ها وجود دارد. به این دلیل که ممکن است نیمی از منابع سیستم پیش از لغو **Restart** بسته شده باشند.

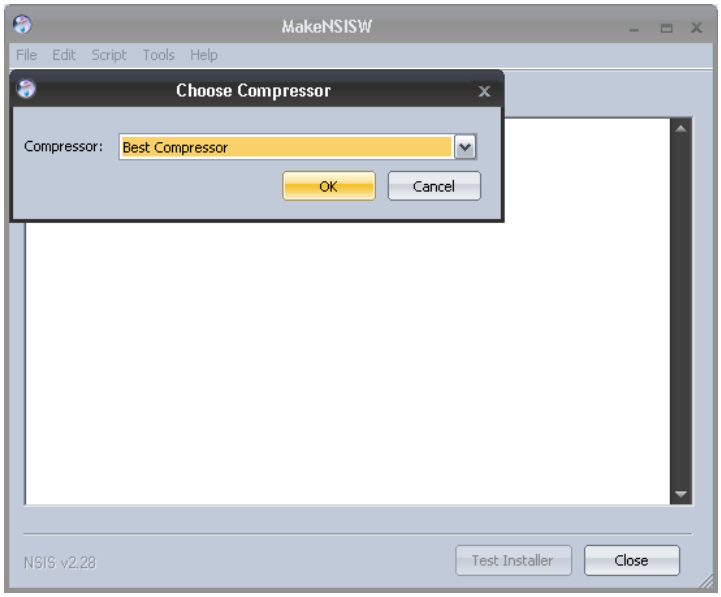
الف-۲-۷-un.onUserAbort

لغو عملیّات برنامه توسّط کاربر باعث اجرای دستورات این تابع می‌شود.

پیوست ب: بهینه سازی

ب-۱-کاهش بیش از پیش حجم

به گونه‌ی کلّی مهم‌ترین عامل افزایش حجم، انتخاب نادرست فشرده‌ساز جاری است. انتخاب فشرده‌سازی مناسب نیاز به دانش و تجربه دارد. بهترین روش برای کسب تجربه، آزمایش همگی حالات مختلف فشرده‌سازی بر روی انواع مختلف فایل‌هاست. برای انجام این کار روی فایل اسکریپت خود کلیک راست کنید و عنوان «**Comile NSIS Script (Choose Compressor)**» را انتخاب کنید. رابط کامپایلر **NSIS** ظاهر می‌شود و از شما گونه‌ی فشرده‌ساز را می‌پرسد.



در لیست مورد نظر گزینه‌ی آخر یعنی **Best Compressor** را انتخاب کنید. با انجام این کار اسکریپت شما با انواع فشرده‌سازهای موجود فشرده می‌شود و کم‌حجم‌ترین نصب به عنوان خروجی تعیین خواهد شد.

یکی از قابلیّت‌های **NSIS** فشرده‌سازی یک‌تکّه (**Solid**) است. به این معنی که همه‌ی فایل‌های موجود در بسته را با هم یکی می‌کند و سپس فایل مورد نظر را فشرده می‌کند. انجام این کار نیاز به فضای خالی به اندازه‌ی همه‌ی فایل‌ها بر روی دیسک سخت است. به علاوه با انجام این کار امکان استخراج مستقیم یک فایل وجود ندارد و همه‌ی فایل‌های موجود پیش از فایل مورد نظر باید پردازش شوند.

برای بهینه کردن فشرده‌سازی یک تکّه از پارامتر **/x** دستور **File** استفاده می‌کنیم. به این صورت که فایل‌های با محتوای مشابه را در امتداد هم قرار می‌دهیم تا فشرده‌سازی در بیشترین حدّ خود قرار بگیرد.

فرض کنید مجموعه‌ای از فایل‌ها در بسته‌ی ما موجود است که شامل **exe**، **txt**، **ocx**، **bmp**، **dll**، **ini**، و... می‌شود.

در حالت ساده ما از دستور **File** به شکل زیر استفاده می‌کنیم.

```
File /r "myApp\*.*"
```

برای مرتّب کردن فایل‌ها در بسته به صورت زیر عمل می‌کنیم.

```
File /r "myApp\*.exe"
File /r "myApp\*.dll"
File /r "myApp\*.ocx"
File /r "myApp\*.bmp"
File /r "myApp\*.txt"
File /r "myApp\*.ini"
File /r "myApp\*.*" /x "*.exe" /x "*.dll" /x "*.ocx" /x "*.bmp" /x "*.txt"\
                    /x "*.ini"  # فشرده کردن بقیه‌ي فایل‌ها
```

همان گونه که در مثال بالا مشاهده می‌کنید فایل‌هایی با محتوای مشابه در امتداد هم قرار گرفتند و در انتها نیز همه‌ی فایل‌ها به جز فایل‌هایی که قبلا فشرده شده بودند در بسته گنجانده شدند.
پارامتر **/x** باعث حذف فایل‌های مورد نظر از لیست فشرده سازی می‌شود.
فراموش نکنید که برای عملکرد بهینه‌ی تکنیک فوق حتماً باید پارامتر **/SOLID** را به دستور **SetCompressor** ارسال کنید.

```
SetCompressor /SOLID lzma
```

از دیگر عواملی که باعث کاهش حجم خروجی می‌شود می‌توان به آیکون مورد استفاده اشاره کرد. یک آیکون می‌تواند شامل چندین تصویر باشد. سعی کنید از آیکون‌هایی با حجم کمتر و فقط شامل تصاویر مورد نیاز خود استفاده کنید.

برای کاهش حجم فایل مجوّز خود بهتر است از فایل متنی معمولی (با پسوند **txt**) استفاده کنید و چنانچه قصد استفاده از **rtf** را دارید، برای ایجاد آن فقط از برنامه‌ی **WordPad** ویندوز استفاده کنید زیرا خروجی آن چندین برابر کوچکتر از فایل مشابه در برنامه‌ای مانند **Word** مجموعه‌ی **Office** است.

چنانچه از رابط مدرن **NSIS** استفاده می‌کنید و تصاویر آن را تغییر داده‌اید حتما با کمک یک ویرایشگر تصویری تعداد رنگ‌های مورد استفاده در تصویر را تا حدّ ممکن کاهش دهید و خروجی را به صورت ۸ بیتی یا ۲۵۶ رنگ ذخیره کنید تا هنگام فشرده‌سازی به بهترین نتایج دست پیدا کنید.

فایل‌های فشرده‌شده چنانچه مجدّد فشرده شوند، کاهش حجم نخواهند داشت و حتّی ممکن است حجمشان افزایش پیدا کند. بنابراین سعی کنید از قرار دادن فایل‌های فشرده در بسته خودداری کنید. در صورت نیاز به فایل فشرده می‌توانید آن‌ها را فشرده‌سازی مجدّد کنید (البتّه در حالت **Store** و بدون فشرده‌سازی).
با انجام همین تکنیک ساده می‌توان حجم تعاریف ویروس‌یاب **Norton** را تا ۲۰ درصد کاهش داد.

فایل‌های کدگذاری شده نسبت به فایل معمول خود بسیار کمتر فشرده می‌شوند. بنابراین چنانچه نیاز به کدگذاری فایل‌ها باکمک پلاگین‌ها را دارید، ابتدا آن‌ها را با کمک یک ابزار جانبی مانند **lzma** فشرده کنید و سپس اقدام به کدگذاری و قراردادن آن در بسته نمایید.
در چنین شرایطی استفاده از دستور **SetCompress** با پارامتر **off** توصیه می‌شود. فراخوانی این دستور پیش از فایل‌های فشرده باعث عدم فشرده‌سازی آن‌ها می‌شود. این دستور هنگام استفاده از فشرده‌سازی یک‌تکّه عملکردی ندارد.

```
SetCompress off
File "myFile.rar"
SetCompress auto
```

ب-۲-افزایش سرعت فشرده سازی

سرعت و قدرت فشرده‌سازی با یکدیگر رابطه‌ی عکس دارند. بنابراین انتخاب با شماست که تا چه اندازه کفه را به سمت هر یک متمایل کنید. برای مثال در انتقال فایل از راه اینترنت، حجم خروجی مهمترین عامل است و سرعت فشرده‌سازی از اهمّیّت کمتری برخوردار است.
سرعت فشرده‌سازیِ انواع فشرده‌سازهای **NSIS** با یکدیگر متفاوت است. بیشترین سرعت را **zlib** در اختیار شما می‌گذارد. به علاوه **zlib** در مورد فایل‌های فشرده بهتر از **bzip2** و **lzma** عمل می‌کند.
استفاده نکردن از پارامتر **SOLID/** در مقابل دستور **SetCompress** نیز باعث افزایش سرعت فشرده‌سازی می‌گردد.

یکی از قابلیّت‌های منحصر به فرد **NSIS** که باعث پیدا کردن قطعات مشابه در فایل‌ها و ذخیره‌ی آن‌ها فقط یک بار می‌شود **DataBlockOptimize** نام دارد. خاموش کردن این بررسی تا حدّ زیادی در هنگام فشرده‌سازی فایل‌های متعدّد سرعت را افزایش می‌دهد. برای خاموش کردن این گزینه در پهنه‌ی سراسری از دستور زیر استفاده کنید.

```
SetDataBlockOptimize off
```

NSIS به گونه‌ی پیش‌فرض همه‌ی فایل‌ها را فشرده می‌کند و چنانچه حجم فشرده‌شده‌ی آن‌ها کمتر از حجم معمول باشد، فایل فشرده‌را در بسته ذخیره می‌کند. با کمک دستور **SetCompress** که روش استفاده از آن را در بخش قبل مشاهده کردید، می‌توان از فشرده کردن مجدّد فایل‌های فشرده جلوگیری کرد و سرعت را تا حدّ زیادی افزایش داد.

استفاده از حجم دیکشنری کوچکتر در مورد فشرده‌ساز **lzma** باعث افزایش سرعت این فشرده‌ساز می‌شود. حجم دیکشنری پیش‌فرض برای **lzma** ۸ مگابایت است. با کم کردن این مقدار می‌توانید اندکی قدرت فشرده‌سازی را کاهش داده و در برابر به سرعت بیفزایید.

سرعت فشرده‌سازی رابطه‌ی مستقیم با قدرت پردازشگر شما دارد. در هنگام فشرده‌سازی سعی کنید تا حدّ امکان برنامه‌های دیگر را ببندید. با کمک **Task Manger** ویندوز می‌توانید فرآیندهایی که پردازش بالایی را مصرف می‌کنند و از اجزای حیاتی ویندوز نیستند را شناسایی کنید وببندید.

درست همانند سرعت پردازنده، سرعت درگاه‌های ورودی و خروجی نیز می‌توانند در کاهش سرعت کامپایل تأثیر گذار باشد. بنابراین سعی کنید در هنگام کامپایل برنامه‌هایی که از فضاهای ذخیره‌سازی و درگاه‌ها استفاده می‌کنند را ببندید.

یک ویروس‌یاب می‌تواند سرعت فشرده‌سازی را تا چند برابر کاهش دهد. پس فراموش نکنید که پیش از شروع کامپایل حتماً ویروس‌یاب خود را ببندید. یک فایروال نا مناسب هم ممکن است تا حدّی در کاهش سرعت تأثیرگذار باشد. در چنین حالتی اینترنت خود را قطع کنید و سپس فایروال را ببندید و فرآیند کامپایل را آغاز کنید.

ب-۳-افزایش سرعت استخراج

هر چه مقدار فشرده‌سازی کمتر باشد سرعت استخراج افزایش خواهد یافت. بنابراین سعی کنید فقط فایل‌های فشرده‌نشده را فشرده کنید (با کمک دستور **SetCompress** که در بخش‌های قبلی شرح آن گذشت).

استفاده از فشرده‌ساز **zlib** بیشترین سرعت را در فشرده‌سازی و استخراج در اختیار شما می‌گذارد و در برابر، نسبت حجم خروجی به ورودی بیشتر خواهد بود.

عوامل بالا هنگامی اثر بخش خواهند بود که فرآیند استخراج از محلّی پر سرعت انجام شود. چنانچه سرعت انتقال داده‌ها پایین است (برای مثال یک شبکه‌ی کم سرعت)، بالاتر بردن قدرت فشرده‌سازی تاثیر مثبت بر سرعت خواهد داشت زیرا حجم داده‌هایی که باید از مسیر آهسته انتقال یابد، کاهش پیدا می‌کند.

با خاموش کردن پارامتر **Overwrite** می‌توانید از استخراج بی‌دلیل فایل‌هایی که قبلاً بر روی مقصد موجود هستند جلوگیری کنید و در نتیجه سرعت استخراج را در حالت خاص افزایش دهید.

برای انجام این کار از دستور **SetOverwrite** استفاده می‌کنیم. این دستور پارامترهای مختلفی را می‌پذیرد:

- **on** – حالت پیش‌فرض – جایگزینی همه‌ی فایل‌ها.
- **off** – عدم جایگزینی در صورت وجود فایل در مقصد.
- **try** – تلاش برای جایگزینی و عدم نمایش پیام در صورت بروز خطا.
- **ifnewer** – بررسی تاریخ دو فایل و جایگزینی در صورت جدیدتر بودن.
- **ifdiff** – جایگزینی در صورت تفاوت داشتن دو فایل.

```
SetOverwrite off
File "myDllFile.dll"
SetOverwrite ifdiff
#ادامه‌ي دستورات
```

همان گونه که پیش‌تر نیز گفته شد، با استفاده از دستور **CRCCheck** می‌توانید بررسی نخستین نصب را که برای اطمینان از درستی بسته انجام می‌شود، خاموش کنید. برای توضیحات بیشتر به بخش ۲-۸-۹ مراجعه کنید