**תקציר**

הרצאה על הכנת חבילות RPM.
חבילות RPM הן דרך מקובלת להפצת תוכנה. בהרצאה זו נראה איך לבנות חבילות כאילו בצורה עצמאית, וחשוב מכך: איך לשנות חבילות קיימות.


**שקף 1**

# הכנת חבילות RPM


צפריר כהן

tzafrir@technion.ac.il


**שקף 2**

## תוכן ענינים



לפתוח חבילה ב-mc

שקף 3

## איך זה נכתב

מסמך זה הוכן, כמובן, על מערכת לינוקס.

שקפים אילו הוכנו ע"י תוכנת LaTeX (בעזרת התמיכה העברית לLaTeX2e)

הקבצים נערכו בעורך VIM, עם קצת עזרה מהחבילה TKTeX, אותה אפשר להוריד מ: http://psl-proxy.technion.ac.il/TKTEX

שקף 4

## א׳ מהן חבילות RPM?

מערכת RPM (Redhat Package Manager) היא מערכת ניהול החבילות שהומצאה ע"י חברת רד-האט ע"מ לייעל את הפצת וניהול התוכנה.

למערכת זו כמה מטרות:

- התקנה והסרה פשוטה של רכיבי תוכנה
- אפשרות לעדכן תוכנה מותקנת
- פישוט ואוטומציה של תהליך בניית ההפצה

בשל היכולות הרבות של מערכת RPM בחרו גם הפצות אחרות (לדוגמה קלדרה ו-SuSE) להשתמש בה.

חשוב להבדיל בין שני דברים: הפורמט RPM, הספריה rpmlib ובסיס הנתונים שהיא מחזיקה על המערכת מחד, לעומת התוכנית rpm, שהיא ממשק משתמש לאותה מערכת, ומטפלת גם בהורדת קבצים, בניית חבילות, ועוד.

שקף 5

## מבנה חבילה בינארית - הארכיון

ישנם שני סוגי חבילות: חבילת מקור, אשר מיועדת להעברת קוד המקור של התוכנית, וחבילה "בינארית", אשר מכינה קבצים מוכנים להתקנה.

עיקרה של חבילת RPM הוא ארכיון cpio המכיל את הקבצים.

ואם כל מה שמעניין אתכם הוא הקבצים, אפשר להשתמש בתוכנית rpm2cpio כדי לקבל ארכיון cpio רגיל מהקבצים. לדוגמה: רשימת הקבצים בארכיון RPM:

```
rpm2cpio tetex-latex-heb-1.0.2-2.noarch.rpm |cpio -t
```

וקבלת הקבצים עצמם:

```
rpm2cpio tetex-latex-heb-1.0.2-2.noarch.rpm |cpio -id
```

שקף 6

## מבנה חבילה בינארית (המשך)

מלבד הקבצים עצמם החבילה צריכה לכלול עוד אינפורמציה רבה. בתחילת החבילה יש מעין טבלה שבה רשומים פרטים כגון:

- שם החבילה
- תאור החבילה
- גרסה
- ארכיטקטורה
- מה החבילה מספקת
- מה החבילה דורשת
- סקריפטים הנדרשים לרוץ בזמן ההתקנה או ההסרה

## יכולות ותלויות

יש חבילות אשר צריכות דברים שחבילות אחרות מספקות. אילו הן הדרישות (**requiremnets**) של החבילה.

לדוגמה: סקריפטים בפרל ידרשו קיום של תוכנית /usr/bin/perl , וכן ייתכן שהם יצטרכו גם כמה מודולים לא סטנדרטיים.

דוגמה יותר טיפוסית היא תוכנית C/C++ אשר משתמשת בספריות כלש-הן ואינה מקושרת סטאטית. במקרה זה התוכנית לא תרוץ אם לא יהיו במערכת הגרסאות המתאימות של הספריות הנ"ל.

מערכת RPM לא יכולה להבטיח את קיום התנאים הללו במערכת הלינוקס עצמה. במקום זה, היא מסתמכת על "הבטחות" של חבילות: כל חבילה מצהירה אילו יכולות (**capabilities**) היא מספקת (**provides**).

שקף 7

## שימוש בסיסי ב-rpm - התקנה והסרה

הפקודה i: התקנת חבילה. האופציה v - verbose

rpm -iv *rpm_file_name*

הפקודה U: עדכון חבילה (מתקין אם אינה מעודכנת). בעדכון נמחקות חבלות ישנות יותר שהיו מותקנות, וכן נשמרים קובצי הקונפיגורציה הקי-ימים (אם הם שונו).

rpm -Uv *rpm_file_name*

הפקודה e - מחיקת חבילה מותקנת. חבילה יכולה להיות מזוהה ע"י שמה בלבד, או ע"י השם המלא, הכולל מספר גרסה (שימושי במקרה של כמה חבילות עם אותו שם בסיסי):

```
rpm -e xboing
rpm -e xbiong-2.4-8mdk
```

שקף 8

## שימוש בסיסי ב-rpm - שאילתות

מערכת RPM מחזיקה מאגר מידע על כל החבילות המותקנות במערכת. בין השאר אפשר לברר שם לאיזו חבילה שייך כל קובץ. כמוכן אפשר לברר על כל חבילה פרטים כגון: שם, תאור, גירסה, רשימת קבצים, וזמן התקנה.

את כל אותם פרטים (למעט זמן התקנה) אפשר לשאול גם על קובץ RPM שאינו בהכרח מותקן.

שקף 9

הגירסה של perl שמותקנת:

```
rpm -q perl
```

מידע כללי על החבילה hdate-5.01-1.i586.rpm:

```
rpm -qpi hdate-5.01-1.i586.rpm
```

כל התוכניות מהחבילה של ps:

```
rpm -qfl `which ps` |grep /bin/
```

## מהלך התקנת חבילות

מה קורה כאשר מבקשים מ-rpm להתקין קבוצת חבילות?

שקף 10

1. בדיקת תקינות: האם מותר להתקין את כל החבילות (ביחד!)
2. לכל חבילה בתורה:
   (2.2) מחיקת חבילות "נדרסות", אם ישנן
   (2.2) הרצת סקריפט pre
   (2.2) העתקת הקבצים מהארכיון למערכת הקבצים
   (2.2) עדכון מאגר המידע של RPM
   (2.2) הרצת סקריפט post

שקף 11

## ב׳ מהן חבילות מקור?

חבילת מקור -- source RPM, היא די דומה בצורתה לחבילה בינארית, אולם בארכיון שלה יש:

- קובץ SPEC -- הוראות הכנת חבילה
- קובצי המקור (בד"כ בארכיוני tar)
- אוסף פאטצ'ים.

מקובל שחבילת המקור כוללת את הארכיון המקורי מה-upstream (מתחזק החבילה המקורית). כל שינוי שרוצים להכניס מתבצע ע"י הוספת פאטצ'. מטרת העיקרון הזה היא להקל על עדכון החבילה כאשר מתקבל עדכון מה-upstream.

שקף 12

## שלבי בניית החבילה

השלבים העיקריים של בניית חבילה עם rpm:

**prep** : פרישת כל הקבצים. מקביל ל-`tar xvzf; patch`

**build** : בניית החבילה עצמה. מקביל ל-`configure; make`

**install** : "התקנה" של החבילה שנוצרה (לעץ זמני תחת תיקיה זמנית)

לאחר שלב ה-install מתבצעת האריזה עצמה: קבצים מהעץ הזמני מועתקים לארכיון, וכן המילון נבנה.

כל שלב חייב לעבור בשלום כדי להגיע לשלב הבא

## בנייה אוטומטית - rpm –rebuild

שקף 13

הפקודה:

```
rpm --rebuild package.src.rpm
```

פותחת את החבילה, בונה אותה (prep, build, install), אורזת חבילה בינרית, ומוחקת את תיקיית הבניה.

אפשר להשתמש בפקודה הזו אם רוצים לבנות מחדש חבילה קיימת (כדי לקבל חבילה בינרית שתתאים לכל הספריות של המערכת הנוכחית,ו/או כדי לבנות חבילה עם אופטימיזציות הקומפיילר הרצויות.

## בניה ידנית

שקף 14

כאשר רוצים לשנות את ההגדרות של החבילה או לעדכן את אחד מקובצי המקור, לא מספיק להשתמש ב-rpm –rebuild. מה שצריך לעשות הוא:

```
rpm -i package.src.rpm
```

כעת החבילה נפרשה, ואפשר לשנותה (פרטים בהמשך). לאחר שגמרנו לשנות מריצים:

```
rpm -ba $RPM/SPEC/package.spec
```

(אפשר להשתמש ב-bb- כדי לבנות רק את החבילה הבינארית)

## בניה ידנית(המשך)

לחילופין, אם רוצים לנסות רק חלקים מסויימים של תהליך בניית החבילה:

הרחבת כל המקרואים בקובץ ה-spec בלבד:

rpm -bE $RPM/SPEC/*package*.spec

ביצוע שלב ה-prep בלבד: שקף 15

rpm -bp $RPM/SPEC/*package*.spec

ביצוע שלבי ה-prep וה-build:

rpm -bc $RPM/SPEC/*package*.spec

ביצוע שלבי ה-prep ה-build, וה-install:

rpm -bi $RPM/SPEC/*package*.spec

## בניה ידנית(המשך)

בדיקת רשימת הקבצים בלבד: שקף 16

rpm -bl $RPM/SPEC/*package*.spec

אפשר להשתמש באופציה short-circuit– בשביל לדלג על כמה שלבים, אבל רק כאשר מריצים bc- או bi-!

## ג׳ קובץ ה-spec

שקף 17

קובץ ה-spec הוא קובץ אשר מכיל את ההוראות להכנת החבילה. הוא הקדמה (*premable*) וכמה חלקים:

**description** : תאור מילולי קצר של החבילה (פסקה אחת או יותר)

**prep, build, install, clean** : החלקים הללו כוללים חלק מסקריפט. התוכנית rpm תשלים את הסקריפט הזה לסקריפט מלא שיבוצע בשלב בעל אותו השם.

**files** : רשימת הקבצים בחבילה, ומאפייניהם

**Changelog** : לא לשכוח לעדכן!

## קובץ ה-spec: ה-premable

שקף 18

לדוגמה:

```
%define name hdate
%define version 5.01
%define release 1
Summary: hdate - hebrew calendar equivalents of date and cal
Name: %{name}
Version: %{version}
Release: %{release}
Source0: ftp://ftp.ivrix.org.il/pub/ivrix/src/cmdline/
calendar/hdate-%{version}.tgz
Copyright: GPL
```

## קובץ ה-spec: ה-premable(המשך)

שקף 19

```
Group: System Environment/Base
BuildRoot: %{_tmppath}/%{name}-buildroot
Prefix: %{_prefix}
URL: ftp://ivrix.org.il/pub/ivrix/src/cmdline/calendar/
#Provides: hdate
Provides: hcal
```

יש כאן שימוש נרחב במקרואים. חלקם (כגון name) מוגדרים בקובץ הזה, אולם חלקם (כגון _tmppath) מוגדרים ע"י "המערכת".

החבילה הזו לא דורשת requires מפורש אף יכולת מהמערכת. בזמן יצירת החבילה יתווספו כמה דרישות בצורה אוטומטית.

## קובץ ה-spec: שלב ה-prep

שקף 20

עבור חבילה פשוטה, עם מקור אחד וללא אף פאטץ', כל התוכן של התוכן של prep% הוא המקרו setup%, אשר פותח את קובץ ה-tar. שימו לב כי b- suffix. הוא קצת קיצור של האופציות המתאימות של patch:

```
%setup
%patch0 -p1 -b .flex
```

אופציה שימושית למקרו setup% היא n dir_name-, אשר מנחה את המערכת להניח שארכיון ה-tar לא מכיל תיקיה אחת בעלת השם name-version אלא תיקיה אחת בעלת השם dir_name

שקף 21

## קובץ ה-spec: ‏%build ול-%install

```
%build
./configure --prefix=%{prefix}
make prefix=%{prefix} DEFS="$RPM_OPT_FLAGS"
%install
#rm -rf $RPM_BUILD_ROOT
#install -d $RPM_BUILD_ROOT%{prefix}/bin
#install -d $RPM_BUILD_ROOT%{_mandir}/man1
make prefix=$RPM_BUILD_ROOT%{prefix} install
%clean
rm -rf $RPM_BUILD_ROOT
```

שקף 22

## קובץ ה-spec: ‏%files

בחלק זה מופיעה רשימה של כל הקבצים שיהיו שייכים לחבילה

- צריך לקבוע גם בעלים והרשאות על הקבצים (הבעלות היא לפי שם, לא לפי u/gid
- אפשר להשתמש ב-* ואף להגדיר תתי-ספריות שלמות כשייכות לחבילה. ה-install
- ‏%doc: עבור קובץ ללא path הקבצים ילך לתקיית התיעוד. עם path - הקובץ מסומן כקובץ תיעוד
- אפשר לסמן קבצים כקובצי תצורה ע"י %config
- גם כאן אפשר ורצוי להשתמש במקרואים כגון %_mandir

שקף 23

# ד׳ נושאים נוספים

שקף 24

## שמירת חבילת מקור כארכיון TAR

אם החבילה כוללת רק ארכיון tar אחד, ולא כוללת אף פאטצ׳, זה בעצם די מיותר ליצור חבילת SRPM בשבילה. אפשר פשוט לשים את קובץ ה-spec בתוך ארכיון ה-TAR.

rpm מאפשרת לבנות חבילת RPM מתוך קובץ TAR שבו יש תיקיה אחת, ובה יש קובץ spec בשם המתאים. הדבר מאפשר למי שמפיץ חבילה כארכיון TAR לאפשר בניה של חבילות RPM מארכיון זה.

במקרה זה משתמשים בפקודה מקבילה ל-rpm –rebuild

```
rpm --rebuildtar package.tar.gz
```

בנוסף לכל פקודת rpm -b*x* יש פקודה מקבילה rpm -t*x* אשר מקבלת קובץ TAR במקום קובץ spec, מוציאה את קובץ ה-spec מקובץ ה-TAR, ופועלת כמו פקודת b- המקבילה. לדוגמה:

```
rpm -tc package.tar.gz
```

שקף 25

## עריכת קובצי RPM ב-Emacs

ב-?? אפשר למצוא גם לינק להורדת rpm-spec-mod.el והוראות התקנה.

mode זה מאפשר הוספה של חלקים שונים לקובץ ה-spec, וכן הרצה של פקודות rpm -b שונות. מומלץ.

כמוכן ViM מגיע עם הדגשת תחביר עבור קובצי SPEC של RPM.

שקף 26

## עבודה כמשתמש רגיל

במהלך הכנת חבילת RPM מורצים הרבה סקריפטים שנוצרים אוטומטית מקוד שכותב המשתמש. זה מצב אשר מזמין שגיאות, ולכן היה רצוי לא לעבוד כ-root.

?? מה צריך לעשות. כדי לעבוד כמשתמש רגיל.

לנוחות הקורא אני מצרף את הדרוש כסקריפט home_dir_rpm

שקף 27

## רשימת מקורות

[1] RPM-HOWTO

[2] Mandrake RPM mini-HOWTO

[3] Maximum RPM