

Linux Development Tools

ברוכים הבאים ללינוקס
כלי פיתוח בלינוקס
שחר שמש

החלטות החלטות החלטות

עולם התוכנה החופשית מלא באפשרויות. בד"כ יש יותר מדרך אחת לעשות את אותה הפעולה.

משתמשים חדשים בד"כ נבוכים משלל האפשרויות.

בהרצאה הזו נציין שמות של חלק מהאופציות, אולם נציג לעומק רק אחת.

אינה בהכרח הטובה ביותר.

אתם מוזמנים לבחון אחרות

הדרך בה לא נלך

הדיון הנצחי – שורת הפקודה כנגד כלים גרפיים. 🐧

ייתרונות כלי שורת הפקודה: 🐧

יכולת התאמה רבה יותר 🐧

מהירים יותר לשימוש – מרגע שעברנו את מחסום הלימוד 🐧

הקשה יותר

סטנרטיים יותר 🐧

בלינוקס וביוניקס – הכלים הגרפיים הינם מעטפות
לכלי שורת הפקודה. 🐧

מאחורי הקלעים - Make

שומר רשימת תלויות, כדי לבצע בניה מינימלית בכל פעם. 🐧

תלויות נשמרות בקובץ בשם "Makefile". 🐧

הפעלה משורת הפקודה: "make". 🐧

תיעוד מלא: 🐧

http://www.gnu.org/software/make/manual/html_node/make_toc.html

מאחורי הקלעים – autoconf

מבצע בדיקות בזמן קומפילציה של תכונות המערכת עליה מהדירים את התוכנה. 🐧

מאפשר לכתוב תוכנה עבור מספר סביבות, תוך התאמות לכל סביבה. 🐧

מקבל קובץ "configure.ac" או "configure.in". מייצר קובץ "configure" אותו מריצים לפני הידור ראשוני. 🐧

תיעוד מלא: 🐧

[http://www.gnu.org/software/autoconf/manual/
autoconf-2.57/autoconf.html](http://www.gnu.org/software/autoconf/manual/autoconf-2.57/autoconf.html)

מאחורי הקלעים - automake

מייצר קבצי Makefile באופן אוטומטי. 🐧

קובץ הכניסה ל-automake הינו קטן מאוד. 🐧

מייצר קבצי make שמכילים את כל מה שצריך, כולל
clean ו-install.

בפועל – השקול ל"קובץ פרוייקט" בעולם החלונאי. 🐧

תיעוד מלא: 🐧

<http://sources.redhat.com/automake/automake.html>

נקודת הפתיחה – סביבת הפיתוח

anjuta, kdevelop, eclipse, gnat, widestudio 🐧

vi (gvim), emacs 🐧

gambas (basic), idle (Python) 🐧


אילו רק הכלים החופשיים. קיימים גם כלים קנייניים 🐧


שארית ההרצאה תתמקד בעיקר ב-KDevelop. 🐧


שימו לב. ההרצאה הינה על גרסה 3. גרסה 2 הינה הרבה יותר פרימיטיבית, ולא מומלצת לעבודה. 🐧

Kdevelop 3 – מדריך מהיר

יצירת פרוייקט חדש – Project/New 

בחירת סוג הפרוייקט. אנחנו נתמקד בפרוייקטי C פשוטים. 

התוכנה מייצרת את כל הקבצים הדרושים לפרוייקט
תוכנה חופשית, כולל רשיון. אם אתם לא צריכים
אותם, פשוט תתעלמו מהם (ללחוץ "next"). 

המערכת מייצרת קבצי automake, autoconf
לפרוייקט שלנו באופן אוטומטי. כתוצאה מכך הידור
ראשוני מעט איטי. 

מורה נבוכים בסביבה

רשימת הקבצים בפרוייקט נמצאת מצד שמאל למטה
– File Tree.

ייצירת קובץ חדש באמצעות “File/New” וציון סוג
הקובץ תבטיח שהקובץ שייך לפרוייקט.

ניהול הקבצים השייכים לפרוייקט מתבצע מצד ימין
למטה - “Automake manager”.

GDB – מנפה השגיאות

קיימים מספר מעטפות ל-GDB 🐧

DDD 🐧

KDBG 🐧

ניתן להפעיל את הדיבאגר גם ישירות מתוך פרוייקט
Kdevelop. 🐧

כל היכולות הסטנדרטיות של מנפה שגיאות: 🐧

break point 🐧

מעקב אחר משתנים 🐧

נקודת עצירה מותנית 🐧

מיפוי משתני התוכנית – CTags

זוהי תוכנה המבצעת מיפוי של כל משתני התוכנית, והיכן הם מוגדרים. 🐧

המיפוי הוא מיפוי ע"פ ניתוח סטטי. 🐧

ניתן לתשאל את הנתונים ע"י הלשונית CTags בתחתית המסך. 🐧

יש ליצור מסד מידע מעודכן לפני השאילתה (כפתור "Regenerate"). 🐧

קבלת עזרה

ישנם מספר מקורות לתייעוד. 🐧

חלק מהתייעוד נמצא המערכת שנגישה ישירות לחיפוש. 🐧

חלק נמצא בדפי תיעוד (Manual pages) ובדפי מידע (info pages). 🐧

דפי ההסבר מחולקים לשמונה מחלקות (Sections). לכל מחלקה יש קטגוריה משלה. 🐧

כאשר אנחנו מקבלים כמה תשובות לאותה מילת חיפוש, ניתן לבחור את התשובה הנכונה ע"י בחירת המחלקה שמתאימה למה שאנחנו מחפשים. 🐧

ניתן לחפש בכולם מתוך KDevelop. 🐧

קבלת עזרה – דפי הסבר

מחלקות דפי ההסבר הן: 

1. פקודות משתמש
2. פקודות מערכת הפעלה (ליבה)
3. פקודות ספריה
4. התקנים
5. תיעוד מבני קבצים
6. משחקים
7. אחר
8. ניהול מערכת

כלי עזר לפיתוח – ניהול תצורה

שומר את ההיסטוריה שהתוכנה בפיתוח עברה. 🐧

מי שחושב שהוא לא צריך – טועה. 🐧

הכלי הנפוץ ביותר כיום – CVS 🐧

מעטפת גרפית – cervisia 🐧

הפעלה מתוך KDevelop יחסית קשה. מומלץ לעבוד
מבחוץ בשלב ראשון. 🐧

Concurrent Versions System – CVS

פעולות חד פעמיות

Repository – מקום בו נשמרים קבצי התצורה בפועל. 🐧

מקומי – במערכת הקבצים המקומית. 🐧

מרוחק – דרך פרוטוקול רשת כלשהו. 🐧

פעולות בסיסיות: 🐧

INIT – ייצירת Repository מקומי. 🐧

IMPORT – הכנסה ראשונית של קבצים קיימים ל-
Repository. 🐧

Check Out – קבלת העתק של קבצים לצורך עבודה. 🐧

Concurrent Versions System – CVS

פעולות שוטפות

עדכון מבנה הקבצים לפי שינויים שקרו במקביל
במצבור – Update.

שמירת השינויים שבוצעו בחוץ אל תוך המצבור –
Commit או Check in.

תיוג של גרסה – Tag.

הצמדת עץ פיתוח לתג מסויים – Stickiness.

כלים מתקדמים – Valgrind

כלי זמן ריצה למציאת שגיאות. 🐧

מריץ את התוכנה בתוך Emulator. 🐧

מוודא שמשתנים מאותחלים לפני השימוש. 🐧

מוודא שזיכרון שניגשים אליו הוקצה כראוי. 🐧

אחרי סיום ריצת התוכנית, מנסה לוודא שכל הזיכרון שהוקצה אכן שוחרר. 🐧

כלים שימושיים

Kompare – משווה שני קבצים. 🐧

splint – מבצע ניתוח סטטי של קובץ C ומצביע על בעיות. 🐧

RATS – מבצע ניתוח סטטי של קובץ C ומצביע על בעיות אבטחת מידע פוטנציאליות. 🐧

Profiling – זיהוי צווארי בקבוק. 🐧

gprof – מבצע ניתוח על תוכנה ספציפית. 🐧

oprofile – מבצע ניתוח של כל המערכת בבת אחת. 🐧