

מעבדה מתקדמת בשפת C

תרגיל מספר 2

תזכורת: חובה להגיש תרגילים בזוגות, למעט מקרים שאושרו פרטנית מול סגל הקורס. התרגילים נבדקים אוטומטית בעזרת מערכת לזיהוי העתקות, כולל מול הגשות מסמסטרים קודמים.

מועדי ההגשה: (ראו הנחיות בסוף קובץ זה לגבי השלבים השונים)

שלב 1: 23.4, 23:59

שלב 2: 30.4, 23:59

גירסה סופית: 7.5, 23:59

התרגיל:

בתרגיל זה נכתוב תוכנית בשם `my_grep` המקבלת כקלט ביטוי ושם קובץ. אם התוכנית לא קיבלה שם קובץ, היא קוראת קלט מ-`stdin`. התוכנית סורקת את הקובץ שורה-שורה ומדפיסה רק את השורות המכילות את הביטוי.

פרמטרים אותם התוכנית יכולה לקבל (התוכנית יכולה לקבל כמה מהפרמטרים ביחד):

`-A NUM` - התוכנית תדפיס גם `NUM` שורות אחרי השורה המכילה את הביטוי. בין בלוקים של שורות רצופות שמודפסות תודפס שורה המכילה שני מקפים בלבד.

`-b` - לפני כל שורה יודפס מספר הבתים מתחילת הקובץ ועד תחילת השורה. שורות שמודפסות עקב שימוש באופציה `-A` יודפסו עם מקף בין מספר השורה לתוכן השורה. שורות שהיו מודפסות גם ללא האופציה הזו, יודפסו עם נקודותיים בין מספר השורה לתוכן השורה.

`-c` - במקום להדפיס את השורות, יודפס רק מספר השורות.

`-i` - התוכנית תתעלם מההבדל בין אותיות גדולות וקטנות.

`-n` - הדפס לפני כל שורה את מספרה בקובץ הקלט. שורות שמודפסות עקב שימוש באופציה `-A` יודפסו עם מקף בין מספר השורה לתוכן השורה. שורות שהיו מודפסות גם ללא האופציה הזו, יודפסו עם נקודותיים בין מספר השורה לתוכן השורה.

`-v` - הדפס רק את השורות שלא מכילות את הביטוי.

x- הדפס רק שורות שמכילות את הביטוי, ולא מכילות דבר פרט לבטוי.

הביטוי אותו מנסה התכנית למצוא יכול להכיל אותיות גדולות וקטנות, ספרות, וכן את כל התווים הנראים הרגילים כלומר כל התווים בטבלת ASCII בתחום 33 עד 126 (דצימלי).

חפוש התווים ().\|{}| יוברח ע"י התו באקסלאש, '.\'. בתרגיל שלנו, הופעה של אחד מהתווים הללו בביטוי (כשהם מוברחים, או כשהם לא מוברחים) מחייבת שימוש במתג E-, ומחייבת שהביטוי יהיה מוקף במרכאות. (גרפ האמיתי תומך באוסף רחב הרבה יותר של קלטים, ועל כך בהמשך.)

בנוסף ניתן יהיה לעשות שמוש בביטויים רגולרים בעזרת הארגומנט E-, שחייב להופיע מיד לפני הביטוי, כאשר הביטוי מוקף במרכאות. הביטוי עצמו יכול להיות מורכב כמפורט:

(str1|str2) – משמעו חפש את הבטוי str1 או str2

. (נקודה) – כל תו יותאם. לא יכולות להופיע נקודות בתוך סוגריים.

[x-y] – מתאים לכל תו בין x ל-y כולל.

נקודות וסוגריים מרובעים ועגולים לא יכולות להופיע בתוך סוגריים מרובעים או עגולים.

דוגמאות:

הביטוי \[0-9](1|) מתאים לכל המספרים הקטנים מעשרים (כולל אפס) המוקפים בסוגריים מרובעים.

הרצת 2013.html "o\pdf" -i -n my_grep על הקבץ הבא:

```
<HTML>
<HEAD>
<TITLE>Yuval Shavitt (Advanced Lab in C)</TITLE>
</HEAD>
<body>
  <h1>Advanced Laboratory in C - 2013</h1>
  <ul>
    <li> <a href="C-intro.pdf">Revisiting C</a>
```

 Modularity
 I/O and files in C

תדפיס:

8: Revisiting C
10: I/O and files in C

באילו קלטים צריך לתמוך, ומה צריך להיות הפלט עליהם?

1. **באילו קלטים צריך לתמוך?** אתם לא צריכים לתמוך באוסף הקלטים החוקיים המלא של גרפ, אלא רק במה שכתוב בהוראות הנ"ל. ודאו שאתם מבינים את אוסף הקלטים בהם אתם נדרשים לתמוך, ושאתם לא משקיעים זמן יקר בקלטים בהם לא צריך לתמוך. דוגמה פופולרית לקלט שלא צריך לתמוך בו (אבל לצערי סטודנטים מבזבזים עליו זמן) הוא הופעה של אחד מהתאים ().\{\} ללא המתג E- או ללא מרכאות. גרפ האמיתי תומך בזה, אבל אתם לא צריכים לתמוך בזה.

2. **מה צריך להיות הפלט?** בקלטים בהם כן צריך לתמוך, ההתנהגות המצופה מהקוד שלכם זהה להתנהגות של גרפ האמיתי, תמיד. הריצו grep בנובה וראו מה ההתנהגות המצופה.

3. **אני לא מבין למה גרפ האמיתי מדפיס מה שהוא מדפיס על קלט מסוים?** אנא שאלו בפורום: רישמו באופן מסודר את שורת הפקודה שאתם מריצים, את הפלט המודפס ואת הפלט שאתם חושבים שגרפ האמיתי צריך להדפיס, והסבירו למה אתם מצפים שיודפס הפלט שאתם מצפים לו. לפני כן, מומלץ גם לחפש בגוגל (או להריץ בנובה) man grep כדי להגיע לתייעוד.

הערות לגבי הקוד:

1. אין לכתוב לקבצים במהלך התוכנית שלכם. כלומר האסטרטגיה של לטפל ב-stdin ע"י כתיבה של התוכן שלו לקובץ "ומשם כבר פתרנו", לא חוקית. האסטרטגיה הזו לא עובדת במצב שאין לכם הרשאות כתיבה.
2. אין להשתמש בפונקציות ספריה המתעסקות בביטויים רגולריים, כגון regcomp וכו'.
3. כל פעולה שאתם עושים בקוד, צריכה להתבצע במקום אחד בדיוק. בפרט, אין טעם לממש שתי פונקציות שבודקות התאמה, אחת לביטוי רגולרי ואחת לביטוי פשוט, אם הפונקציה של הביטוי הרגולרי יכולה לטפל גם בביטוי פשוט. מצב כזה מהווה שכפול קוד, כי הוא גורר את כל הבעיות שקורות כשמשכפלים קוד, כפי שדיברנו בכיתה. הוא גם גורר הורדת נקודות.

נהלי הגשה

תחת התיקיה `c_lab` שיצרתם לטובת הגשת התרגיל הקודם, יש ליצור תיקיה בשם `ex2`. יש לשים בתיקיה תת-תיקיות כמתואר מטה עם קבצי הקוד וקובץ ההרצה, ליצור בכל תת-תיקיה עותק של קובץ ה-`README` מהתרגיל הקודם, ולוודא שיש להם הרשאות 755. יש לשמור את התרגיל המוגש באותה תיקיה עד לסיום הקורס - ייתכן שתידרשו להשתמש בו בתרגילים שיגיעו בעתיד.

פיתוח אינקרמנטלי

במסגרת התרגיל, נתרגל פיתוח בשיטה אינקרמנטלית:

- עד מועד ההגשה של שלב 1 (ראו בראש התרגיל) צריכה להיות בתיקיית התרגיל תת תיקיה בשם `beta1` העוברת בהצלחה את הטסטים המורצים ע"י הפקודה:
`~nimrodav/grep_tests/beta1.sh`
- עד מועד ההגשה של שלב 2 (ראו בראש התרגיל) צריכה להיות בתיקיית התרגיל תת תיקיה בשם `beta2` העוברת בהצלחה את הטסטים המורצים ע"י הפקודה:
`~nimrodav/grep_tests/beta2.sh`
- עד מועד ההגשה של הגירסה הסופית (ראו בראש התרגיל) תיקיית התרגיל הראשית צריכה לעבור את כל הטסטים המורצים ע"י הפקודה
`~nimrodav/grep_tests/run_all.sh`

יש להריץ את הפקודות בתיקיה (או תת-תיקיה) בה נמצא קובץ ההרצה בעזרת השורות הרשומות מעלה. אם הפקודה לא הדפיסה אף שורה למסך, הכל בסדר. אם היא הדפיסה שורות למסך, הריצו את כל אחת מתת-הבדיקות שהיא מריצה ובידקו מה לא תקין (עשו `cat` לקובץ הזה כדי לראות מה הן תת הבדיקות). הפקודה בודקת שההרשאות של כל הקבצים בתיקיה, ושל התיקיה עצמה, הן בדיוק 755. אנא ודאו שאלה בדיוק ההרשאות, כי הפקודה תדפיס למסך כל קובץ שההרשאות שלו אינן 755. מומלץ לוודא גם ידנית במועד ההגשה שכל ההרשאות הן אכן 755. הערה: בדיקת התרגיל, כולל תיקיות ה-`beta`, תיעשה במועד ההגשה הסופי שלו. הבדיקה תוודא שתיקיות ה-`beta` לא עודכנו לאחר מועדי ההגשה שלהן.

ימי חסד לגירסאות הבטא

לצורך הגשת גירסאות הבטא, עומדים לרשותכם סה"כ ארבעה ימי חסד נפרדים, כלומר איחורים בגירסאות הבטא לא יורדים ממכסת ימי החסד הרגילים. איחור בהגשת הגירסה הסופית של התרגיל, כן יורד ממכסת ימי החסד הרגילים.

שימוש ב-clang-format, clang-tidy, valgrind

גירסת ההגשה של התרגיל צריכה להיות נקייה מאזהרות של valgrind ו-clang-tidy, ואחרי שהקוד עבר פירמוט בעזרת clang-format. גירסאות הבטא גם הן צריכות להיות נקיות מאזהרות של valgrind, אך לא חייבות להיות נקיות מאזהרות של clang-tidy, ולא חייבות להיות לאחר פירמוט בעזרת clang-format. הטסט האוטומטי של התרגיל אוכף זאת.

שימוש ב-Makefile

יש ליצור Makefile, התומך בקימפול של כל מודול בנפרד, כפי שלמדנו בכיתה. ה-Makefile צריך גם לתמוך במטרות test, all ו-clean, שהן מסוג PHONY. מטרת test מריצה את קובץ הטסט של התרגיל. שימו לב: make עשויה לא להכיר את התו תילדה ~ עבור תיקיית בית. ניתן לתת נתיב מלא אל קובץ הטסט כפי שהוא יושב בתיקיה שלי, או להעתיק אותו אליכם ולהריץ אותו מתוך התיקיה שלכם.

קובץ DESIGN

בגירסת ההגשה הסופית, יש ליצור בתיקיה קובץ בשם DESIGN, המכיל שורה עבור כל מודול. כל שורה נפתחת בשם המודול, וכוללת משפט שמסביר מה תוכן המודול. שימו לב: אם מילות המפתח בהסבר לא מופיעות בשם המודול, זה סימן שאולי שם המודול לא מתאים.

מעבר נוסף על המסמך "סדר פעולות בדיקת קובץ H" ועל ההוראות

אחרי שסיימתם לכתוב את הקוד ולפני הגשת התרגיל, אנא קיראו בשנית את המסמך "סדר פעולות בדיקת קובץ H" שמופיע במודל, ו-ודאו שקבצי ה-H שלכם עומדים בקריטריונים שהוא מגדיר.

לנוחותכם, ועל מנת להקל על שיתוף הפעולה בין שותפים, אנא צרו קובץ בשם SDP בתיקיית ההגשה, אשר מכיל שורה אחת ובה רצף האותיות SDP. כך תוכלו להיות בטוחים שלפחות אחד מהשותפים עבר על הקובץ הנ"ל, ושלא תאבדו נקודות סתם. הטסט האוטומטי של התרגיל אוכף זאת.

כמו כן, יש ליצור בתיקיה קובץ בשם UNIFORM_MATCHING, על מנת להבהיר שקראתם את הערה מס' 3 הנ"ל, לגבי זה שאין טעם לממש שתי פונקציות התאמה. כמו כן, יש ליצור קובץ נוסף בתיקיה בשם COPYING_WILL_BE_REPORTED, שמכיל שורה אחת ובה שם הקובץ, COPYING_WILL_BE_REPORTED. זאת על מנת להבהיר לי וגם לכם שאתם מבינים שמקרים של העתקה ידווחו לוועדת משמעת.

בהצלחה!