Ohjelmointikielet edistyneet piirteet -kurssin pieni projekti: mygrep

Tämän noin kolmen viikon pituisen pienen projektin ideana on koostaa yhteen kaikki Erkin kurssiosuudella opittu oppi sekä aiempi oppi C++ -ohjelmoinnista.

Ideana on tehdä pienehkö mygrep-niminen ohjelma, jonka esikuva on Linux:sta löytyvä grep-työkalu. Grep:ssä ominaisuuksia riittää mutta meidän ohjelma keskittyy ihan sen ytimeen. Grep-ohjelman käyttöohje löytyy tämän dokumentin perästä. Saman löydät Linux:sta komennolla ”man grep” tai vaikka samoilla termeillä google-hakuna. Grep-työkalulla voit etsiä tiedostosta rivit jotka sisältävät haluamasi merkkijonon.

Työ tehdään inkrementteinä joista ensimmäisen tekeminen takaa yhden pisteen projektista; toisen kolme pistettä, kolmannen neljä pistettä ja neljän inkrementin tekeminen antaa kaikki viisi pistettä. Projekti arvioidaan siis pistemäärällä 0 – 5 pistettä ja sen vaikutus kokonaisarvosanaan on kerrottu toisaalla. Inkrementit tulee tehdä järjestyksessä alusta alkaen ja kaikkien inkrementtien tulee toimia siihen inkrementtiin asti jota vastaavaa pistemäärää tavoittelet. Esim. viiden pisteen ohjelmasta tulee löytyä yhden pisteen ohjelman toiminnot. Yhdestä pisteestä hypätään suoraan kolmeen sen vuoksi että työmäärässä tapahtuu myös vähän isompi nousu tuossa kohtaa.

Suurin lisäarvo tämän projektin tekemisellä on siinä, että projektissa sovelletaan käytäntöön opittuja asioita jonkin verran isomman ohjelman tekemisessä. Työmäärä asettuu samaan kuin jos tekisi kurssilla kolmen viikon ajan kaikki kotitehtävät. Älkää kuitenkaan olko huolissanne työmäärästä; kaikki eteneminen vie teidän osaamistanne eteenpäin!

Inkrementit on kerrottu alla. Alta löytyy myös ajoesimerkkejä. Lopusta löytyy vaatimukset sille mitä sinun pitää palauttaa tästä projektista saadaksesi pisteet.

Tästä speksistä poikkeamista ei sallita; opettaja testaa teidän ohjelmia ja ajaa mahdollisesti automaattitestejä ohjelmiinne joten jos käyttöliittymänne ei ole tämän speksin mukainen testit eivät onnistu ja asiakas ei hyväksy ohjelmaanne (näin toimitaan ohjelmistoyrityksissä asiakastilanteissa). On hyvä jo tässä vaiheessa törmätä siihen asiaan että mitä vaatimusmäärittelyssä lukee on se asia mistä asiakas maksaa jatkossa (jos ne vaatimukset on tehty).

1. Inkrementti (takaa yhden pisteen)

Ohjelma käynnistetään komentoriviltä alla listatulla tavalla. Ajoesimerkki on otettu Mac-tietokoneelta; voit ajaa ohjelmaa samalla tavalla myös

Windows 10:n cmd-työkalussa. Tässä ajettavan ohjelman (binääri) nimi on mygrep ja ./mygrep tarkoittaa sitä että se käynnistetään nykyisestä työhakemistosta:

```
./mygrep
Give a string from which to search for: Erkki Hietalahti
Give search string: rkki

"rkki" found in "Erkki Hietalahti" in position 1
```

Alla toinen ajo:

```
./mygrep
Give a string from which to search for: Erkki Hietalahti
Give search string: rkkk

"rkkk" NOT found in "Erkki Hietalahti"
```

Kuten esimerkeistä näkyy ohjelma kysyy käyttäjältä ”ison” merkkijonon josta etsitään annettua pienempää merkkijonoa. Jälkimmäinen voi löytyä isosta merkkijonosta mistä kohtaa tahansa. Jos etsittävä merkkijono löytyy isosta merkkijonosta sen esiintymiskohta kerrotaan positiosta 0 alkaen. Kaikissa tapauksissa tulostetaan ilmoitus löytymisestä.

Periaatteessa voit ajaa tätä ohjelmaa myös käyttämäsi IDE-välineen (vaikkapa Microsoft Visual Studio) valikosta tai pikanäppäinyhdistelmän avulla mutta jatkossa ohjelmaa kannattaa ajaa komentoriviltä koska sen käyttäminen perustuu oleellisesti komentoriviargumenttien antamiseen ja niitä on IDE-välineellä tyypillisesti kömpelö asettaa.

2. Inkrementti (tämä ja 1. inkrementti takaavat kolme pistettä jne.)

Kuten aiemmin sanottiin tämän inkrementin tulee sisältää myös edellisen inkrementin toiminnallisuuden eli jos edelleen ohjelma käynnistetään pelkällä ohjelmatiedoston (binääri) nimellä se toimii kuten edellisessä luvussa on kerrottu.

Mutta jos ohjelma käynnistetään alla kerrotulla tavalla komentoriviltä sen käyttö alkaa muistuttamaan perus-grep -käyttöä komentoriviargumentteineen. Testaa siis ohjelmaasi tämän jälkeen suoraan komentoriviltä.

Alla ajoesimerkki:

```
./mygrep following man_grep_plain_ASCII.txt
     The following options are available:
     The grep utility exits with one of the following values:
```

Ohjelma käy lävitse sille viimeisenä annetun ASCII-muotoisen tiedoston sisällön rivi riviltä ja jos kulloisenkin rivin sisältä löytyy jostain kohtaa haettu merkkijono (2. komentoriviargumentti) ko. rivi tulostetaan; muutoin ei. Toiminta on täsmälleen saman kaltaista kuin grep:in toiminta sen yksinkertaisimmassa käyttötilanteessa.

Annan teidän käyttöönne tekstitiedoston man_grep_plain_ASCII.txt joka sisältää nimensä mukaisesti grep:in manuaalisivun raakana ASCII-tekstinä. Voitte käyttää sitä ohjelmanne testaamisessa; teidän ohjelman tulee löytää samalla tavoin nuo esimerkin kaksi tekstiriviä mitkä näkyvät yo. tulostuksessa jos annatte ohjelmallenne samat komentoriviargumentit ja käytätte samaa man_grep_plain_ASCII.txt -tiedostoa.

3. Inkrementti (kaikki tähänastiset inkrementit 3. inkrementti ml. takaavat neljä pistettä)

Tähän inkrementtiin on poimittu mallina muutama grep:in tekemä yksinkertainen tilastollinen analyysi. Eli sinun mygrep:iä pitää pystyä ajamaan niin että:

- Haluttaessa se tulostaa löydettyjen rivien eteen niiden rivinumerot isossa tiedostossa
- Haluttaessa ajon lopussa mygrep tulostaa tiedon siitä montako riviä tiedostossa oli joista löytyi hakemasi merkkijonon esiintymä.

Näitä molempia ominaisuuksia säädetään komentoriviargumenttien avulla: jos rivinumerointi halutaan siitä ilmoitetaan komentoriviargumentilla; sama pätee rivien lukumäärän laskentaan.

Alla esimerkki ajamisesta:

```
./mygrep –olo following man_grep_plain_ASCII.txt
32:     The following options are available:
245:     The grep utility exits with one of the following values:

Occurrences of lines containing "following": 2
```

Edellisessä tulostuksessa tulostettujen rivien numerot näkyvät rivien edessä; man_grep_plain_ASCII.txt -tiedoston rivillä 32 on täsmälleen tulostuksessa näkyvä sisältö. Samoin löytyneitä rivejä on kaksi kappaletta.

Kuten näet ajoesimerkistä nämä ylimääräiset valinnaiset optiot tarjotaan muodossa -olo. Voit tulkita tämän tarkoittavan sitä että ”-o” kertoo että sen perässä luetellaan haluttuja ohjelman ajoon vaikuttavia optioita (engl. options) ja l = rivinumerointi (engl. line numbering) ja o = löytyneiden rivien lukumääräoptio (engl. occurrences). Nyt komentoriviargumentteja on neljä kolmen sijaan.

Jos optiomääritys on -ol vain rivinumerot tulostetaan ja jos se on -oo vain esiintyneiden rivien lukumäärä tulostetaan. Käyttäjä voi antaa siten näitä valikoiden.

Huom: kun tarjoat tätä inkrementtia opettajalle tarkastukseen myös edellisten inkrementtien pitää toimia!

4. Inkrementti (kaikki tähänastiset inkrementit 4. inkrementti ml. takaavat viisi pistettä)

Tähän inkrementtiin on upotettu seuraavat lisävaatimukset edellisten inkrementtien vaatimusten lisäksi:

- Ohjelmassa pitää soveltaa jossain kohtaa poikkeuskäsittelyä. Luontainen kohta sen soveltamiseen on tiedostokäsittely: onko mainittu tiedosto olemassa, onko siihen lukuoikeus jne. ?
- Haluttaessa merkkijonohaku riviltä tehdään niin että isot ja pienet kirjaimet ovat saman arvoisia. Silloin esim. etsittävät merkkijonot ”erkki”, ”Erkki”, ”eRkKi” ja ”ERKKI” jne. ovat samoja; erkki-merkkijonoa etsitään oli se kirjoitettu millä tavalla tahansa (isoja ja pieniä kirjaimia sekaisin).
- Haluttaessa mygrep ohjelmasi etsiikin ne rivit joilla **ei ole** antamaasi merkkijonoa.

Alta löydät ajoesimerkkejä kustakin vaatimuksesta; niistä näet myös vaaditut optioasetukset komentorivillä.

```
./mygrep -or following man_grep_plain_ASCII.txt

GREP(1)                  BSD General Commands Manual                 GREP(1)

NAME
     grep, egrep, fgrep, zgrep, zegrep, zfgrep -- file pattern searcher

SYNOPSIS
     grep [-abcdDEFGHhIiJLlmnOopqRSsUVvwxZ] [-A num] [-B num] [-C[num]]
          [-e pattern] [-f file] [--binary-files=value] [--color[=when]]
          [--colour[=when]] [--context[=num]] [--label] [--line-buffered]
          [--null] [pattern] [file ...]

DESCRIPTION
     The grep utility searches any given input files, selecting lines that
     match one or more patterns.  By default, a pattern matches an input line

     <… JA PALJON PALJON LISÄÄ RIVEJÄ; ITSE ASIASSA KAIKKI RIVIT POISLUKIEN NE
      KAKSI JOISTA LÖYTYY SANA following …>
```

Optiomäärittelyssä r tarkoittaa käänteistä etsimistä eli etsitään rivejä joilla **ei** ole haettua merkkijonoa (r = engl. reverse search).

Toinen ajoesimerkki:

```
./mygrep -oi folloWING man_grep_plain_ASCII.txt
     The following options are available:
     The grep utility exits with one of the following values:
```

Optio i tarkoittaa että isot ja pienet kirjaimet ovat samanarvoiset (i = engl. ignore case). Siten folloWING löytyy kyllä riviltä jonka osana on teksti following.

Kolmas ajoesimerkki jossa on kaikki optiot päällä:

```
./mygrep -olori folloWING man_grep_plain_ASCII.txt
1:
2:GREP(1)                BSD General Commands Manual                GREP(1)
3:
4:NAME
5:     grep, egrep, fgrep, zgrep, zegrep, zfgrep -- file pattern searcher
6:
7:SYNOPSIS
8:     grep [-abcdDEFGHhIiJLlmnOopqRSsUVvwxZ] [-A num] [-B num] [-C[num]]
9:        [-e pattern] [-f file] [--binary-files=value] [--color[=when]]
10:       [--colour[=when]] [--context[=num]] [--label] [--line-buffered]
11:       [--null] [pattern] [file ...]
12:
13:DESCRIPTION
14:     The grep utility searches any given input files, selecting lines that
15:     match one or more patterns.  By default, a pattern matches an input line

<… TÄSSÄ VÄLISSÄ ON PALJON TULOSTUVIA RIVEJÄ RIVINUMEROINEEN …>

Occurrences of lines NOT containing "folloWING": 299
```

-olori tarkoittaa: tulostetaan rivinumerot (l), tulostetaan niiden rivien lukumäärä joista ei löytynyt (o), tehdään käänteinen etsiminen eli etsitään rivit joilta merkkijonoa ei löydy (r), ei tehdä eroa isoille ja pienille kirjaimille (i).

Neljäs ajoesimerkki jossa näkyy poikkeuskäsittely:

```
mv man_grep_plain_ASCII.txt man_grep_plain_ASCII.txt2
./mygrep -olori folloWING man_grep_plain_ASCII.txt
An exception occurred. Exception Nr. -1
Could not find out the size of file "man_grep_plain_ASCII.txt"
```

Ohjelmassa on siis varauduttu siihen että sille annetaan parametrina olematon tiedosto. Ensimmäinen kohta missä yllä esitelty esimerkkiohjelma käsittelee ko. tiedostoa on sen koon selvittäminen tavuina; juuri tämä käsittely epäonnistuu ensiksi tiedoston puuttumisen takia ja siihen on varauduttu poikkeuskäsittelyllä.

Opittujen asioiden soveltaminen

Eräs tavoite tämän projektin tekemisellä on soveltaa Erkin kurssiosuudella esiteltyjä asioita käytäntöön. Tässä muutamia vinkkejä tähän:

- Osoittimia kannattaa käyttää yksittäisten tekstirivien prosessoinnissa kun etsit sieltä haettua merkkijonoa
- Haluttaessa voit lukea koko käsiteltävän tiedoston sisällön dynaamisesta muistista varattuun saman kokoiseen muistialueeseen (= char-taulukko). Tämän jälkeen kaikki

merkkijonoetsintään liittyvät toiminnot ovat tämän muistialueen selaamista lävitse. Voit tehdä sitä osoittimien avulla.
- Parametrien välittämiseltä et voi välttyä; tässä projektissa kannattaa tehdä ehdottomasti aliohjelmia ja käyttää niissä fiksuja tapoja välittää tietoja erilaisten parametrien ja paluuarvon avulla
- Tiedostonkäsittely on oleellinen osa tätä projektia
- Poikkeusmekanismia käytetään 5 pistettä tuovassa inkrementissa.

Väkisin ei pidä käyttää opittua asiaa siellä minne se ei sovi tai käyttö tuntuu keinotekoiselta. Valitsemme aina optimaalisimmat tekniikat kussakin ohjelmointitilanteessa.

Mitä palautetaan

Tehdystä projektista palautetaan moodleen sinne osoitettuun paikkaa todisteet siitä että projekti on tehty. Alla tarkemmat vaatimukset:

- Raportti josta ilmenee:
  - Ohjelman suunnitteluratkaisu pseudokoodina, kaaviona tai ihan sanallisesti: mistä osista ohjelma rakentuu, mitä käsittelyvaiheita se sisältää, jos ohjelmasta löytyy haastava algoritmi kuvataan se pseudokoodina tai vastaavalla esitystavalla
  - Todisteet siitä että ohjelmasi kääntäminen ja linkkaaminen onnistuvat ilman virheitä ja varoituksia. Todisteina toimivat näytönkuvat käyttämistäsi kääntäjistä. Jos käännöksesi on vähänkään monimutkaisempi (ei pelkkä MVS:n (= Microsoft Visual Studio) Build) kerro myös käännöskomento.
  - Todisteet siitä että ohjelmasi inkrementit toimivat. Aja tässä dokumentissa kerrotut ohjelman ajot samoin optioin ja komentiriviargumentein kuin yllä (käytä tarjolla olevaa tekstitiedostoa hakemisen kohteena) ja sinun ohjelmasi tulosten pitää olla samat kuin mitä on listattu tähän dokumenttiin. Lisää raporttiisi näytönkuvat kaikista ajamistasi testeistä ja selittävät tekstit siitä miten näytönkuvat todistavat ohjelmasi oikean toiminnan. Jos teit ohjelmaasi poikkeuskäsittelyn (viiden pisteen ohjelma) sinun tulee näyttää testaustuloksilla sen toimivuus (esim. näytönkuvien avulla).
  - Projektiasiat: paljonko sinulla meni tähän projektiin aikaa päivittäin. Lisää siis tähän kohtaan kirjanpitosi työtunneista tyyliin pvm, työtunnit, mitä noiden tuntien sisällä tehtiin jokaisesta erillisestä työrupeamastasi.
  - Oppiminen: mitä opit tehdessäsi tätä projektia ? Vahvistuiko kurssilla käsiteltyjen asioiden omaksuminen ? Sana on tässä kohden vapaa …
  - Mitkä inkrementit teit ja kuinka montaa pistettä tavoittelet. Huom: 1. inkrementti = 1p, 2. inkrementti (sisältää 1. ja 2.

inkrementit) = 3p, 3. inkrementti (sis. 1. & 2. & 3. inkrementit) = 4p ja viimeinen inkrementti (sisältää kaiken ;) = 5p.
- Yhteystietosi siltä varalta että palautustasi pitää käsitellä enemmän. Vähintään sähköpostiosoite tarvitaan; mielellään myös puhelinnumero.
- Osoite kaikki lähdekoodit sisältävään Git-repoon jos käytit sellaista. Minulla pitää olla siihen lukuoikeus.
- Lisää raportin perään liiteeksi kaikki tekemäsi lähdekoodi tekstinä. Liittäminen pitää tehdä seuraavassa muodossa (esimerkki); liitä tiedostojen sisällöt tiedostojen nimen mukaisessa aakkosjärjestyksessä:

...
fileContents.cpp:
---
otsikossa mainitun lähdetekstitiedoston sisältö

main.cpp:
---
otsikossa mainitun lähdetekstitiedoston sisältö

jne.

Huom: raportti on yksi Word-dokumentti jonka muotoilu on TAMKin raportointiohjeiden mukainen ja se sisältää osinaan näytönkuvia jne. Tee raportille järkevä lukuotsikointi; raportin pitää olla selkeä että tarkastaja ymmärtää sen.

Moodleen palautetaan erikseen projektista tehty raportti ja tehdyt lähdetekstitiedostot (ellet antanut tarkastajalle tiedoksi ne sisältävät git-repositoryä; tarkastajan pitää pystyä lukemaan repositoryn sisältöä!). Molemmat tiedostot nimetään seuraavasti (esim. jos Sauli Niinistö olisi palauttaja!): mygrepProjekti_SauliNiinistö_4inkrementtiä_5pistettä.docx ja mygrepProjekti_SauliNiinistö_4inkrementtiä_5pistettä.zip. Edellinen tiedosto sisältää raportin ja jälkimmäinen tehdyt lähdetekstitiedostot pakattuna. Siten molemmista nimistä pitää selvitä projektin tekijä, tehtyjen inkrementtien määrä ja tavoiteltu pistemäärä projektista.

Liite: grep:in käyttöohje

```
GREP(1)                BSD General Commands Manual                GREP(1)

NAME
     grep, egrep, fgrep, zgrep, zegrep, zfgrep -- file pattern searcher

SYNOPSIS
     grep [-abcdDEFGHhIiJLlmnOopqRSsUVvwxZ] [-A num] [-B num] [-C[num]]
         [-e pattern] [-f file] [--binary-files=value] [--color[=when]]
         [--colour[=when]] [--context[=num]] [--label] [--line-buffered]
         [--null] [pattern] [file ...]

DESCRIPTION
     The grep utility searches any given input files, selecting lines that
     match one or more patterns.  By default, a pattern matches an input line
     if the regular expression (RE) in the pattern matches the input line
     without its trailing newline.  An empty expression matches every line.
     Each input line that matches at least one of the patterns is written to
     the standard output.

     grep is used for simple patterns and basic regular expressions (BREs);
     egrep can handle extended regular expressions (EREs).  See re_format(7)
     for more information on regular expressions.  fgrep is quicker than both
     grep and egrep, but can only handle fixed patterns (i.e. it does not
     interpret regular expressions).  Patterns may consist of one or more
     lines, allowing any of the pattern lines to match a portion of the input.

     zgrep, zegrep, and zfgrep act like grep, egrep, and fgrep, respectively,
     but accept input files compressed with the compress(1) or gzip(1) com-
     pression utilities.

     The following options are available:

     -A num, --after-context=num
             Print num lines of trailing context after each match.  See also
             the -B and -C options.

     -a, --text
             Treat all files as ASCII text.  Normally grep will simply print
             ``Binary file ... matches'' if files contain binary characters.
             Use of this option forces grep to output lines matching the spec-
             ified pattern.

     -B num, --before-context=num
             Print num lines of leading context before each match.  See also
             the -A and -C options.

     -b, --byte-offset
             The offset in bytes of a matched pattern is displayed in front of
             the respective matched line.

     -C[num, --context=num]
             Print num lines of leading and trailing context surrounding each
             match.  The default is 2 and is equivalent to -A 2 -B 2.  Note:
             no whitespace may be given between the option and its argument.

     -c, --count
             Only a count of selected lines is written to standard output.

     --colour=[when, --color=[when]]
             Mark up the matching text with the expression stored in
             GREP_COLOR environment variable.  The possible values of when can
```

be `never', `always' or `auto'.

-D action, --devices=action
Specify the demanded action for devices, FIFOs and sockets. The default action is `read', which means, that they are read as if they were normal files. If the action is set to `skip', devices will be silently skipped.

-d action, --directories=action
Specify the demanded action for directories. It is `read' by default, which means that the directories are read in the same manner as normal files. Other possible values are `skip' to silently ignore the directories, and `recurse' to read them recursively, which has the same effect as the -R and -r option.

-E, --extended-regexp
Interpret pattern as an extended regular expression (i.e. force grep to behave as egrep).

-e pattern, --regexp=pattern
Specify a pattern used during the search of the input: an input line is selected if it matches any of the specified patterns. This option is most useful when multiple -e options are used to specify multiple patterns, or when a pattern begins with a dash (`-').

--exclude
If specified, it excludes files matching the given filename pattern from the search. Note that --exclude patterns take priority over --include patterns, and if no --include pattern is specified, all files are searched that are not excluded. Patterns are matched to the full path specified, not only to the filename component.

--exclude-dir
If -R is specified, it excludes directories matching the given filename pattern from the search. Note that --exclude-dir patterns take priority over --include-dir patterns, and if no --include-dir pattern is specified, all directories are searched that are not excluded.

-F, --fixed-strings
Interpret pattern as a set of fixed strings (i.e. force grep to behave as fgrep).

-f file, --file=file
Read one or more newline separated patterns from file. Empty pattern lines match every input line. Newlines are not considered part of a pattern. If file is empty, nothing is matched.

-G, --basic-regexp
Interpret pattern as a basic regular expression (i.e. force grep to behave as traditional grep).

-H Always print filename headers with output lines.

-h, --no-filename
Never print filename headers (i.e. filenames) with output lines.

--help Print a brief help message.

-I Ignore binary files. This option is equivalent to --binary-file=without-match option.

-i, --ignore-case
Perform case insensitive matching. By default, grep is case sensitive.

--include
If specified, only files matching the given filename pattern are searched. Note that --exclude patterns take priority over --include patterns. Patterns are matched to the full path specified, not only to the filename component.

--include-dir
If -R is specified, only directories matching the given filename pattern are searched. Note that --exclude-dir patterns take priority over --include-dir patterns.

-J, --bz2decompress
Decompress the bzip2(1) compressed file before looking for the text.

-L, --files-without-match
Only the names of files not containing selected lines are written to standard output. Pathnames are listed once per file searched. If the standard input is searched, the string ``(standard input)'' is written.

-l, --files-with-matches
Only the names of files containing selected lines are written to standard output. grep will only search a file until a match has been found, making searches potentially less expensive. Pathnames are listed once per file searched. If the standard input is searched, the string ``(standard input)'' is written.

--mmap Use mmap(2) instead of read(2) to read input, which can result in better performance under some circumstances but can cause undefined behaviour.

-m num, --max-count=num
Stop reading the file after num matches.

-n, --line-number
Each output line is preceded by its relative line number in the file, starting at line 1. The line number counter is reset for each file processed. This option is ignored if -c, -L, -l, or -q is specified.

--null Prints a zero-byte after the file name.

-O If -R is specified, follow symbolic links only if they were explicitly listed on the command line. The default is not to follow symbolic links.

-o, --only-matching
Prints only the matching part of the lines.

-p If -R is specified, no symbolic links are followed. This is the default.

-q, --quiet, --silent
Quiet mode: suppress normal output. grep will only search a file until a match has been found, making searches potentially less expensive.

-R, -r, --recursive
Recursively search subdirectories listed.

```
     -S      If -R is specified, all symbolic links are followed.  The default
             is not to follow symbolic links.

     -s, --no-messages
             Silent mode.  Nonexistent and unreadable files are ignored (i.e.
             their error messages are suppressed).

     -U, --binary
             Search binary files, but do not attempt to print them.

     -V, --version
             Display version information and exit.

     -v, --invert-match
             Selected lines are those not matching any of the specified pat-
             terns.

     -w, --word-regexp
             The expression is searched for as a word (as if surrounded by
             `[[:<:]]' and `[[:>:]]'; see re_format(7)).

     -x, --line-regexp
             Only input lines selected against an entire fixed string or regu-
             lar expression are considered to be matching lines.

     -y      Equivalent to -i.   Obsoleted.

     -Z, -z, --decompress
             Force grep to behave as zgrep.

     --binary-files=value
             Controls searching and printing of binary files.  Options are
             binary, the default: search binary files but do not print them;
             without-match: do not search binary files; and text: treat all
             files as text.

     --context[=num]
             Print num lines of leading and trailing context.  The default is
             2.

     --line-buffered
             Force output to be line buffered.  By default, output is line
             buffered when standard output is a terminal and block buffered
             otherwise.

     If no file arguments are specified, the standard input is used.

ENVIRONMENT
     GREP_OPTIONS  May be used to specify default options that will be placed
                at the beginning of the argument list.  Backslash-escaping
                is not supported, unlike the behavior in GNU grep.

EXIT STATUS
     The grep utility exits with one of the following values:

     0    One or more lines were selected.
     1    No lines were selected.
     >1    An error occurred.

EXAMPLES
     To find all occurrences of the word `patricia' in a file:

          $ grep 'patricia' myfile
```

To find all occurrences of the pattern `.Pp' at the beginning of a line:

```
$ grep '^\.Pp' myfile
```

The apostrophes ensure the entire expression is evaluated by grep instead of by the user's shell. The caret `^' matches the null string at the beginning of a line, and the `\' escapes the `.', which would otherwise match any character.

To find all lines in a file which do not contain the words `foo' or `bar':

```
$ grep -v -e 'foo' -e 'bar' myfile
```

A simple example of an extended regular expression:

```
$ egrep '19|20|25' calendar
```

Peruses the file `calendar' looking for either 19, 20, or 25.

SEE ALSO

ed(1), ex(1), gzip(1), sed(1), re_format(7)

STANDARDS

The grep utility is compliant with the IEEE Std 1003.1-2008 (``POSIX.1'') specification.

The flags [-AaBbCDdGHhIJLmoPRSUVwZ] are extensions to that specification, and the behaviour of the -f flag when used with an empty pattern file is left undefined.

All long options are provided for compatibility with GNU versions of this utility.

Historic versions of the grep utility also supported the flags [-ruy]. This implementation supports those options; however, their use is strongly discouraged.

HISTORY

The grep command first appeared in Version 6 AT&T UNIX.

BUGS

The grep utility does not normalize Unicode input, so a pattern containing composed characters will not match decomposed input, and vice versa.