



Jani Heinikoski 0541122, Vili Huusko 0544718

Harjoitustyöprojektit

Lappeenranta-Lahti University of Technology LUT

CT30A3370 Käyttöjärjestelmät ja systeemiohjelmointi

26.04.2021

Sisällysluettelo

Sisältö

1	Projekti 1: Warmup to C and Unix programming	3
2	Projekti 2: Unix Utilities	5
3	Projekti 3: Unix Shell	10

1 Projekti 1: Warmup to C and Unix programming

Linkki projektiin 1:

https://github.com/ViliAu/CT30A3370_Harjoitustyot/tree/master/Project_1

Projektissa 1 meidän piti rakentaa C-ohjelmointikielellä ohjelma *"reverse"*, joka ottaa CLI-argumentteina joko:

1. Nolla argumenttia
2. Yhden argumentin
3. Kaksi toisistaan eroavaa argumenttia

Ohjelman päätarkoitus on lukea rivi kerrallaan tietystä sisääntulosta ja tulostaa rivit vastakkaisessa järjestyksessä tiettyyn ulostuloon. Tapauksessa 1 (nolla argumenttia), ohjelma lukee standardisääntulosta käyttäjän syötettä, kunnes käyttäjä syöttää pelkän rivinvaihtomerkin `'\n'`. Rivinvaihtomerkin jälkeen ohjelma tulostaa käyttäjän antaman syötteen vastakkaisessa järjestyksessä.

Tapauksessa 2 (yksi argumentti), ohjelma yrittää avata argumenttina annetun tiedoston lukutilassa ja tulostaa sitten tiedoston rivit vastakkaisessa järjestyksessä standardiulostuloon. Kolmannessa tapauksessa, ohjelma yrittää avata ensimmäisenä argumenttina annetun tiedoston lukutilassa ja sitten tulostaa sen rivit vastakkaisessa järjestyksessä toisena argumenttina annettuun tiedostoon.

Miten toteutimme ohjelman?

Pidimme ennen itse ohjelmoinnin aloittamista noin tunnin kestoisen suunnittelusession. Ensin loimme lyhyen projektin vaatimuksista koostuvan vaatimuslistan. Listan pohjalta suunnittelimme, mitä eri osaongelmia projektissa on (tiedoston avaaminen oikeassa tilassa, virheiden käsittely, haarautuminen argumenttien mukaan, rivien kääntämäinen vastakkaiseen järjestykseen jne.). Jaoimme projektia osaongelmiin, kunnes saimme listan riittävän pienistä hallittavista kokonaisuuksista, joiden pohjalta loimme otsikkotiedoston. Otsikkotiedostoon listasimme karkeasti, mitä eri funktioita/proseduureja tarvitsemme koko ohjelman toteuttamiseksi. Tiivistettynä suunnittelimme ja toteutimme ohjelman proseduraalisen ohjelmointiparadigman ominaisen hajota ja hallitse tyylin mukaisesti.

Päädyimme ratkaisemaan ongelman seuraavasti. Ensin avataan haluttu sisääntulo, jota lähdetään lukemaan pinorakenteeseen. Kun sisääntulo on luettu kokonaan muistiin dynaamiseen pinorakenteeseen, niin siirrytään tulostamaan rivit oikeaan ulostuloon vastakkaisessa järjestyksessä. Pino tietorakenteena soveltuu erittäin hyvin tähän, koska pinoon voidaan asettaa vain "päällimmäiseksi" asioita ja poistaa aina vain "päällimmäinen". Tällöin pinon kasaamisen jälkeen riittää purkaa pino tulostaen ja vapauttaen sen alkiot muistista.

Jani Heinikoski toteutti dynaamisen muistinhallinnan (pinorakenteen, sen tulostamisen ja vapauttamisen muistista), Vili Huusko toteutti tiedosto I/O-toiminnot. Molemmat testasimme lopuksi yhdessä Valgrind -ohjelmistolla, että ohjelma ei vuoda muistia missään tapauksessa. Tarkistimme myös lähdekoodin käymällä sitä yhdessä läpi ja korjasimme samalla pieniä loogisia virheitä ja selkeytimme ohjelmakoodin luettavuutta. Varmistimme, että ohjelmakoodi on luettavaa myös lukemalla toistemme toteuttamat osiot, mikäli jokin kohta oli hankala ymmärtää, niin muutimme sen selkeämmäksi.

Kuvakaappaukset toimivasta ohjelmasta, sekä testattavan tiedoston sisällöstä

```
vili@vili-laptop:~/Desktop/Koulu/Kysteemi/CT30A3370_Harjoitustyot/Project_1$ cat test.txt
hello
this
is
a test
```

Kuva 1. Testitiedoston sisältö

```
vili@vili-laptop:~/Desktop/Koulu/Kysteemi/CT30A3370_Harjoitustyot/Project_1$ ./reverse null.txt
error: cannot open file 'null.txt'
```

Kuva 2. Ohjelman ajo olemattomalla tiedostolla

```
vili@vili-laptop:~/Desktop/Koulu/Kysteemi/CT30A3370_Harjoitustyot/Project_1$ ./reverse test.txt
a test
is
this
hello
```

Kuva 3. Ohjelman ajo testitiedostolla

```
vili@vili-laptop:~/Desktop/Koulu/Kysteemi/CT30A3370_Harjoitustyot/Project_1$ ./reverse
reverse
this
is
reversed
from
stdin

stdin
from
reversed
is
this
```

Kuva 4. Ohjelman ajo syötteellä

2 Projekti 2: Unix Utilities

Linkki projektiin 2:

https://github.com/ViliAu/CT30A3370_Harjoitustyot/tree/master/Project_2

Projekti kaksi koostui neljästä pienestä utiliteettiohjelmasta my-cat, my-grep, my-zip, my-unzip. My-cat ohjelma ottaa yhden tai useamman tiedoston CLI-argumenttina ja tulostaa niiden sisällöt standardiulostuloon. Mikäli ohjelmalle ei anneta argumentteja, niin se vain poistuu paluuarvolla nolla.

My-grep ohjelman tarkoitus puolestaan on lukea annetusta sisääntulosta rivi kerrallaan dataa ja tulostaa argumenttina annetun hakutermin sisältävät rivit standardiulostuloon.

My-zip ohjelman tarkoitus on pakata tiedostoja hyödyntäen yksinkertaistettua jakson pituuden koodausta (eng. Run Length Encoding RLE). Ohjelmalle annetaan yksi tai useampi sisääntulotiedosto CLI-argumentteina, jotka ohjelma lukee merkki kerrallaan. Ohjelma pakkaa peräkkäin toistuvat kirjaimet muotoon $k * c$, missä k on 32-bittinen merkitön kokonaisluku, joka ilmaisee, kuinka monta kertaa kirjain c (ASCII-koodattuna) toistui peräkkäin alkuperäisessä tiedostossa. Pakattu data on binäärimuotoista ja se tulostetaan standardiulostuloon, jonka voi ohjata halutessa tiedostoon. Mikäli alkuperäisessä tiedostossa on merkkejä, jotka eivät kuulu 7-bittiseen ASCII-merkistöön, niin ohjelma jättää ne huomiotta, jotta my-unzip ohjelmalle on varmistettu luettavan datan integriteetti.

My-unzip ohjelma kulkee käsi kädessä my-zip ohjelman kanssa, koska sen tarkoitus on palauttaa my-zip ohjelmalla pakattu data alkuperäiseen muotoon. Molemmat ohjelmat käsittelevät vain tekstitiedostoja, joiden sisältö on 7-bittisellä ASCII:lla koodattu.

Miten toteutimme ohjelmat?

Pidimme vastaavasti projektissa kaksi ennen toteuttamista noin tunnin suunnittelusession. Luimme tarkasti, mitä eri ohjelmissa vaaditaan ja jaoimme samalla molemmille kaksi ohjelmaa toteutettavaksi. Jani Heinikoski toteutti ohjelmat my-cat ja my-zip, Vili Huusko loput.

My-cat ohjelma oli kohtuullisen yksinkertainen ja pieni kokonaisuus, joten sitä ei erikseen suunniteltu vaan lähdettiin suoraan toteuttamaan. My-grep puolestaan vaati hieman suunnittelua. Vili Huusko toteutti alustavan suunnitelman ohjelmalle kynällä ja paperilla ja sen pohjalta toteutti sen. My-zip ohjelma vaati myös suunnittelua ja Linuxin man-sivujen tutkimista funktiosta fwrite. Ohjelman suunnittelu tapahtui hajota ja hallitse periaatteella, vastaavasti kuten ensimmäisessä projektissa. My-unzip puolestaan oli

kohtuullisen yksinkertainen eikä vaatinut muuta, kuin fread-funktion man-sivujen tutkimista.

Luimme lopuksi toistemme tekemät ohjelmat, testasimme ne jälleen Valgrindilla ja varmistimme, että molemmat ymmärtävät jokaisen ohjelman toiminnan samalla korjaten ohjelmista löytyneitä bugeja ja epäselkeitä osuuksia.

Kuvakaappaukset toimivista ohjelmista, sekä testitiedostojen sisällöistä

```
vili@vili-laptop:~/Desktop/Koulu/Kysteemi/CT30A3370_Harjoitustyot/Project_2$ cat test.txt
this is
a
test file
aaaaabbbbabab
```

Kuva 5. Testitiedosto 1

```
vili@vili-laptop:~/Desktop/Koulu/Kysteemi/CT30A3370_Harjoitustyot/Project_2$ cat test2.txt
this
is
another
file!
aaaaabbbbbooooo
```

Kuva 6. Testitiedosto 2

```
vili@vili-laptop:~/Desktop/Koulu/Kysteemi/CT30A3370_Harjoitustyot/Project_2$ ./my-cat
```

Kuva 7. My-cat -ohjelman ajo parametreitta

```
vili@vili-laptop:~/Desktop/Koulu/Kysteemi/CT30A3370_Harjoitustyot/Project_2$ ./my-cat nonexistent
my-cat: cannot open file
```

Kuva 8. My-cat -ohjelman ajo olemattomalla tiedostolla

```
vili@vili-laptop:~/Desktop/Koulu/Kysteemi/CT30A3370_Harjoitustyot/Project_2$ ./my-cat test.txt
this is
a
test file
aaaaabbbbabab
```

Kuva 9. My-cat -ohjelman ajo yhdellä tiedostolla

```
vili@vili-laptop:~/Desktop/Koulu/Kysteemi/CT30A3370_Harjoitustyot/Project_2$ ./my-cat  
y-cat test.txt test2.txt  
this is  
a  
test file  
aaaaabbbbabab  
this  
is  
another  
file!  
aaaaabbbbbooooo
```

Kuva 10. My-cat -ohjelman ajo usealla tiedostolla

```
vili@vili-laptop:~/Desktop/Koulu/Kysteemi/CT30A3370_Harjoitustyot/Project_2$ ./my-grep  
y-grep  
my-grep: searchterm [file...]
```

Kuva 11. My-grep -ohjelman ajo ilman parametreja

```
vili@vili-laptop:~/Desktop/Koulu/Kysteemi/CT30A3370_Harjoitustyot/Project_2$ ./my-grep foo  
y-grep foo  
foo bar  
foo bar  
foobar  
foobar  
bafoo  
bafoo  
fobar
```

Kuva 12. My-grep -ohjelman ajo "stdin" -vakiosyötteellä

```
vili@vili-laptop:~/Desktop/Koulu/Kysteemi/CT30A3370_Harjoitustyot/Project_2$ ./my-grep a test.txt  
y-grep a test.txt  
a  
aaaaabbbbabab
```

Kuva 13. My-grep -ohjelman ajo yhdellä tiedostolla

```
vili@vili-laptop:~/Desktop/Koulu/Kysteemi/CT30A3370_Harjoitustyot/Project_2$ ./my-grep a test.txt test2.txt
a
aaaaabbbbabab
another
aaaaabbbbbooooo
```

Kuva 14. My-grep -ohjelman ajo usealla tiedostolla

```
vili@vili-laptop:~/Desktop/Koulu/Kysteemi/CT30A3370_Harjoitustyot/Project_2$ ./my-zip
y-zip
Usage: ./my-zip file1 [file2 ...]
```

Kuva 15. My-zip -ohjelman ajo ilman parametreja

```
vili@vili-laptop:~/Desktop/Koulu/Kysteemi/CT30A3370_Harjoitustyot/Project_2$ ./my-zip test.txt
this is
a
test file
ababab
```

Kuva 16. My-zip -ohjelman ajo "stdout" -vakiotulosteeseen


```

vili@vili-laptop:~/Desktop/Koulu/Kysteemi/CT30A3370_Harjoitustyot/Project_2$ ./my-zip test.txt test2.txt > test.z
vili@vili-laptop:~/Desktop/Koulu/Kysteemi/CT30A3370_Harjoitustyot/Project_2$ xxd -b test.z
00000000: 00000001 00000000 00000000 00000000 01101000 00000001 ....t.
00000006: 00000000 00000000 00000000 01101000 00000001 00000000 ...h..
0000000c: 00000000 00000000 01101001 00000001 00000000 00000000 ..i...
00000012: 00000000 01101011 00000001 00000000 00000000 00000000 .s....
00000018: 00100000 00000001 00000000 00000000 00000000 01101001 ....i
0000001e: 00000001 00000000 00000000 00000000 01101011 00000001 ....s.
00000024: 00000000 00000000 00000000 00001010 00000001 00000000 .....
0000002a: 00000000 00000000 01100001 00000001 00000000 00000000 ..a...
00000030: 00000000 00001010 00000001 00000000 00000000 00000000 .....
00000036: 01101000 00000001 00000000 00000000 00000000 01100101 t....e
0000003c: 00000001 00000000 00000000 00000000 01101011 00000001 ....s.
00000042: 00000000 00000000 00000000 01101000 00000001 00000000 ...t..
00000048: 00000000 00000000 00100000 00000001 00000000 00000000 .. ...
0000004e: 00000000 01100110 00000001 00000000 00000000 00000000 .f....
00000054: 01101001 00000001 00000000 00000000 00000000 01101100 i....l
0000005a: 00000001 00000000 00000000 00000000 01100101 00000001 ....e.
00000060: 00000000 00000000 00000000 00001010 00000101 00000000 .....
00000066: 00000000 00000000 00000000 01100001 00000000 00000000 ..a...
0000006c: 00000000 01100010 00000001 00000000 00000000 00000000 .b....
00000072: 01100001 00000001 00000000 00000000 00000000 01100010 a....b
00000078: 00000001 00000000 00000000 00000000 01100001 00000001 ....a.
0000007e: 00000000 00000000 00000000 01100010 00000001 00000000 ..b..
00000084: 00000000 00000000 00001010 00000001 00000000 00000000 .....
0000008a: 00000000 01101000 00000001 00000000 00000000 00000000 .t....
00000090: 01101000 00000001 00000000 00000000 00000000 01101001 h....i
00000096: 00000001 00000000 00000000 00000000 01100011 00000001 ....s.
0000009c: 00000000 00000000 00000000 00001010 00000001 00000000 .....
000000a2: 00000000 00000000 01101001 00000001 00000000 00000000 ..i...
000000a8: 00000000 01100011 00000001 00000000 00000000 00000000 .s....
000000ae: 00001010 00000001 00000000 00000000 00000000 01100001 ....a
000000b4: 00000001 00000000 00000000 00000000 01101110 00000001 ....n.
000000ba: 00000000 00000000 00000000 01101111 00000001 00000000 ...o..
000000c0: 00000000 00000000 01101000 00000001 00000000 00000000 .t....
000000c6: 00000000 01101000 00000001 00000000 00000000 00000000 .h....
000000cc: 01100101 00000001 00000000 00000000 00000000 01100010 e....r
000000d2: 00000001 00000000 00000000 00000000 00001010 00000001 .....
000000d8: 00000000 00000000 00000000 01100110 00000001 00000000 ...f..
000000de: 00000000 00000000 01101001 00000001 00000000 00000000 ..i...
000000e4: 00000000 01101100 00000001 00000000 00000000 00000000 .l....
000000ea: 01100101 00000001 00000000 00000000 00000000 00100001 e....!
000000f0: 00000001 00000000 00000000 00000000 00001010 00000101 .....
000000f6: 00000000 00000000 00000000 01100001 00000101 00000000 ...a..
000000fc: 00000000 00000000 01100010 00000101 00000000 00000000 ..b...
00000102: 00000000 01101111 00000001 00000000 00000000 00000000 .o....
00000108: 00001010 .

```

Kuva 17. My-zip -ohjelman ajo määritettyyn tiedostoon useammalla tiedostolla kompressoinnin kera

```

vili@vili-laptop:~/Desktop/Koulu/Kysteemi/CT30A3370_Harjoitustyot/Project_2$ ./my-unzip
my-unzip: file1 [file2 ...]

```

Kuva 18. My-unzip -ohjelman ajo ilman parametreja

```

vili@vili-laptop:~/Desktop/Koulu/Kysteemi/CT30A3370_Harjoitustyot/Project_2$ ./my-unzip test.z
this is
a
test file
aaaaabbbbabab
this
is
another
file!
aaaaabbbbbooooo

```

Kuva 19. My-unzip ajo yhdellä parametrilla

```

vili@vili-laptop:~/Desktop/Koulu/Kysteeni/CT30A3370_Harjoitustyot/Project_2$ ./my-unzip test.z > unziptest.txt
vili@vili-laptop:~/Desktop/Koulu/Kysteeni/CT30A3370_Harjoitustyot/Project_2$ cat unziptest.txt
this is
a
test file
aaaaabbbbabab
this
is
another
file!
aaaaabbbbbooooo

```

Kuva 20. My-zip ohjelman kirjoitus tiedostoon

3 Projekti 3: Unix Shell

Linkki projektiin 3:

https://github.com/ViliAu/CT30A3370_Harjoitustyot/tree/master/Project_3

Kolmannen projektin päätarkoituksena oli ymmärtää, kuinka Unixin kuori (ts. komentotulkki) on rakennettu, rakentamalla yksinkertainen Unix kuori ”*wish*”. Kuori piti tehdä siten, että kun sille antaa komennon, niin se luo lapsiprosessin, joka suorittaa komennon ja palauttaa kontrollin käyttäjälle, kun komento on suoritettu.

Kuorelle piti luoda kaksi mahdollista tilaa käyttää sitä; interaktiivinen tila ja batch tila. Interaktiivisessa tilassa kuori antaa kehoitteen (eng. prompt) standardiulostuloon käyttäjälle, jossa hän voi suorittaa komentoja kirjoittamalla niitä suoraan standardisääntuloon. Batch tilassa kuori suorittaa sarjan komentoja batch tiedostosta.

Kuori suorittaa valmiita suoritettavia ohjelmia etsimällä niitä annetusta path muuttujasta. Käyttäjä voi määrittää path muuttujan sisältämät polut suorittamalla kuoren sisäänrakennetun komennon path. Path -komento ottaa nolla tai useamman argumenttia, jotka ylikirjoittavat vanhan path -muuttujan sisällön. Muut sisäänrakennetut komennot ovat exit ja cd. Exit yksinkertaisesti kutsuu funktiota exit(0) eli poistuu kuoresta onnistuneesti (argumenttien antaminen exit -komennolle on virhe). Cd -komento puolestaan ottaa tasan yhden argumentin ja vaihtaa prosessin nykyisen hakemiston hyödyntämällä chdir -järjestelmäkutsua.

Kun kuori suorittaa valmiita ohjelmia (ts. ei sisäänrakennettuja komentoja), niin ohjelmien ulostulon (sekä standardiulostulon, että standardivirheulostulon) voi ohjata tiedostoon syöttämällä ’> <file_name>’ komennon jälkeen.

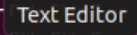
Miten toteutimme ohjelman

Suunnittelimme jälleen ohjelman hajota ja hallitse tyylillä. Ohjelman toteutuksen teimme kokonaan ns. ”pair programming”-tyylillä eli toteutimme ohjelmaa yhdessä pohtien ja koodaten samalla päätelaitteella. Rakensimme ohjelmaa aina pieni pala kerrallaan ja testasimme sitä Valgrindilla, jotta oli aina helppoa rakentaa varmasti toimivan päälle. Aina kun saimme osan testattua, niin pushasimme sen versionhallintajärjestelmään.

Otimme muutamia vapausasteita ohjelman toteuttamisen suhteen. Kuori ei hyväksy syötteenä muuta kuin 7-bittisiä ASCII-merkkejä, jotta syötteen selaaminen olisi edes jotenkin hallittavaa. Mikäli käyttäjä antaa polun esimerkiksi vakiopolku /bin, niin käyttäjän on annettava komento muodossa /ls. Kuori ei myöskään hyväksy rinnakkaisesti suoritettavan **sisäänrakennettuja** komentoja. Mikäli batch tiedostossa on yksikin syntaksivirheen sisältävä rivi (huomaa, että komento, jota ei löydy annetusta hakupolusta ei ole syntaksivirhe), niin suoritusta ei jatketa sen jälkeen.

Kuori ensin jakaa käyttäjän syötteen (tai batch tiedoston rivin) tekstialkioihin (tokeneihin). Tätä vaihetta kutsutaan selaamiseksi. Selaamisen yhteydessä tehdään useita saniteettitarkistuksia, jotka tarkastelevat syötteen syntaksin oikeellisuutta. Mikäli syöte läpäisee tarkistukset, niin kuori haaraantuu sen mukaan, onko syöte sisäänrakennettu komento vai ei. Mikäli komento ei ole sisäänrakennettu, niin se yritetään suorittaa lapsiprosessissa. Sisäänrakennetut komennot suoritetaan isäntäprosessissa.

Kuvakaappaukset toimivasta ohjelmasta

```
vili@vili-laptop:~/Desktop/Koulu/Kysteemi/CT30A3370_Harjoitustyot/Project_3$ ./wish
wish>path
wish>/ls
wish>ls
wish>path /bin/
wish>ls
a.txt out.txt subdir test_batch.txt wish wish.c
wish>ls & ls -la > out.txt
wish>cat --show-all out.txt
a.txt$
out.txt$
subdir$
test_batch.txt$
wish$
wish.c$
total 72$
drwxrwxr-x 3 vili vili 4096 huhti 26 00:37 .$
drwxrwxr-x 6 vili vili 4096 huhti 25 15:25 ..$
-rw-rw-r-- 1 vili vili 40 huhti 26 00:35 a.txt$
- 1 vili vili 34 huhti 25 15:25 .gitignore$
-rw-rw-r-- 1 vili vili 48 huhti 26 00:40 out.txt$
drwxrwxr-x 2 vili vili 4096 huhti 26 00:34 subdir$
-rw-rw-r-- 1 vili vili 112 huhti 26 00:34 test_batch.txt$
-rwxrwxr-x 1 vili vili 23240 huhti 26 00:27 wish$
-rw-rw-r-- 1 vili vili 20391 huhti 25 23:07 wish.c$
wish>
```

Kuva 21. Sisäisen path-komennon käyttö, sekä komentojen ajaminen, sekä uudelleenohjaus tiedostoon

```
wish>ls
a.txt out.txt subdir test_batch.txt wish wish.c
wish>cd subdir
wish>ls
sub_out.txt
wish>echo Hello from a subdir! > sub_out.txt
wish>cat sub_out.txt
Hello from a subdir!
wish>
```

Kuva 22. Sisäisen cd -komennon käyttö, sekä uudelleenohjaus tiedostoon

```
vili@vili-laptop:~/Desktop/Koulu/Kysteemi/CT30A3370_Harjoitustyot/Project_3$ ./wish nonexistent
An error has occurred
Could not open batch file
```

Kuva 23. Vääränlaisen batch -tiedoston lukeminen

```
vili@vili-laptop:~/Desktop/Koulu/Kysteemi/CT30A3370_Harjoitustyot/Project_3$ ./wish test_batch.txt
total 72
drwxrwxr-x 3 vili vili 4096 huhti 26 00:47 .
drwxrwxr-x 6 vili vili 4096 huhti 25 15:25 ..
-rw-rw-r-- 1 vili vili 40 huhti 26 00:35 a.txt
-rw-rw-r-- 1 vili vili 34 huhti 25 15:25 .gitignore
-rw-rw-r-- 1 vili vili 526 huhti 26 00:40 out.txt
drwxrwxr-x 2 vili vili 4096 huhti 26 00:46 subdir
-rw-rw-r-- 1 vili vili 141 huhti 26 00:47 test_batch.txt
-rwxrwxr-x 1 vili vili 23240 huhti 26 00:27 wish
-rw-rw-r-- 1 vili vili 20391 huhti 25 23:07 wish.c
An error has occurred
Chdir failed
An error has occurred
Command not found
vili@vili-laptop:~/Desktop/Koulu/Kysteemi/CT30A3370_Harjoitustyot/Project_3$ cat test_batch.txt
path
path /bin/
ls -la
cd nonexistent
cd ./subdir
ls -la echo this is printed\n from a batch file! > out.txt
THIS COMMAND DOESN'T EXIST
exit
```

Kuva 24. Kuoren ajo batch -tiedostolla, sekä batch-tiedoston sisältö

```
vili@vili-laptop:~/Desktop/Koulu/Kysteemi/CT30A3370_Harjoitustyot/Project_3$ ./wish
wish>path /bin/ ./
wish>gcc wish.c -o wish_new -Wall -ansi -Werror
wish>wish_new
wish>cd subdir
wish>/ls
out.txt sub_out.txt
wish>exit
wish>ls
'&' a.txt out.txt subdir test test_batch.txt test_wish validredir wish wish.c wish_new
wish>exit
```

Kuva 25. Komentojen ajaminen useammilla parametreilla, lapsiprosessien käyttäytymisen havainnollistaminen, sekä useamman polun asettaminen PATH-muuttujaan

```
wish>ls>invalid redir
An error has occurred
Invalid input
wish>ls>validredir
wish>ls&ls&ls&ls&ls&ls>validredir
wish>ls&ls&&
An error has occurred
Invalid input
wish>ls&&ls
An error has occurred
Invalid input
wish>ls & & ls
An error has occurred
Invalid input
wish>ls > &
wish>exit
```

Kuva 26. Erilaisten parsintatapojen havainnollistaminen