Practical Linux Examples

- Processing large text file
- Parallelization of independent tasks

Qi Sun & Robert Bukowski
Bioinformatics Facility
Cornell University

http://cbsu.tc.cornell.edu/lab/doc/linux_examples_slides.pdf
http://cbsu.tc.cornell.edu/lab/doc/linux_examples_exercises.pdf

Estimate the percentage of sequencing reads in the FASTQ file that contains the adapter "AGATCGGAAGAGC".

- **Read file:**

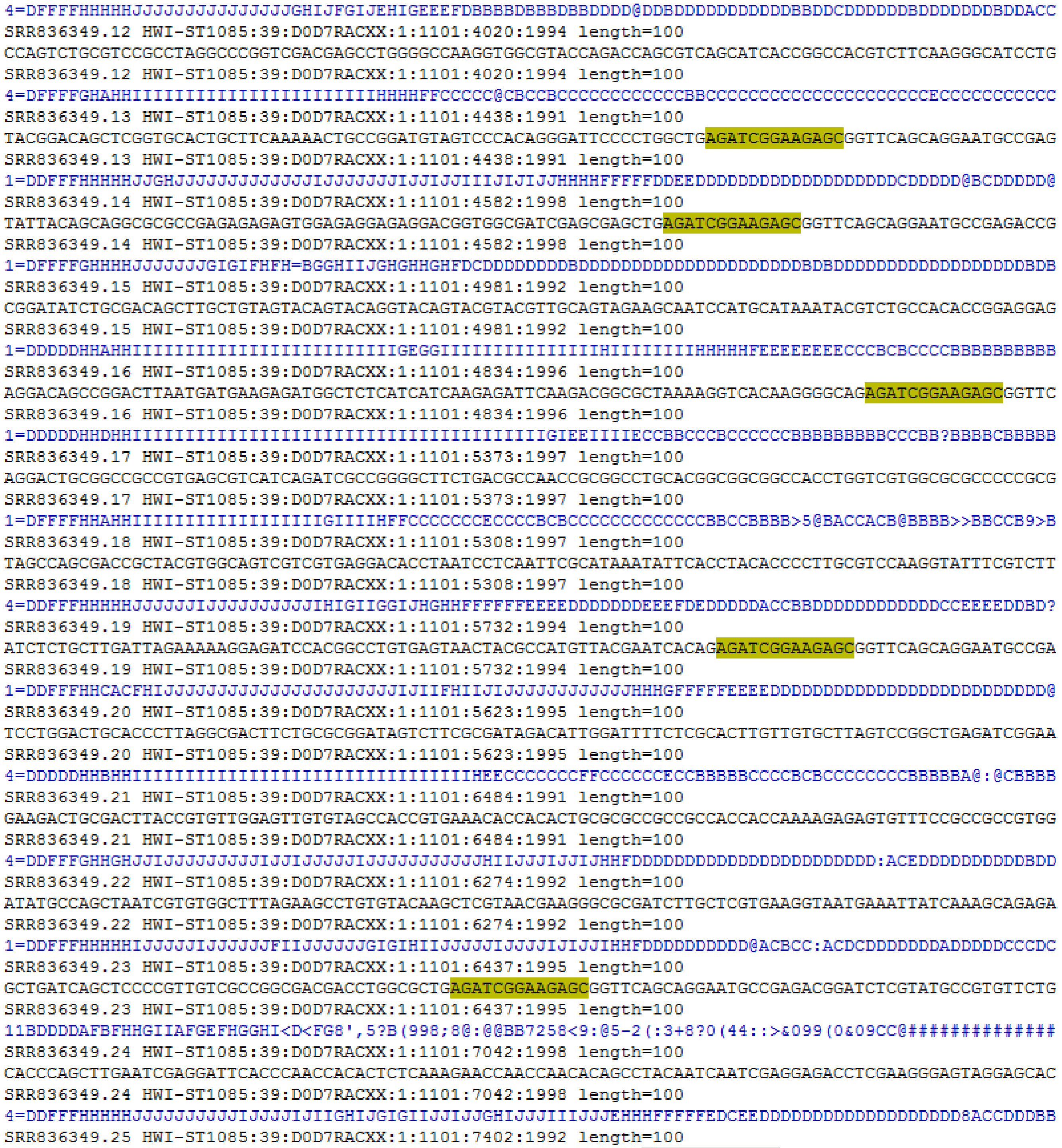
  `cat D7RACXX.fastq`

- **Select lines that contain the string "AGATCGGAAGAGC"**

  `grep AGATCGGAAGAGC`

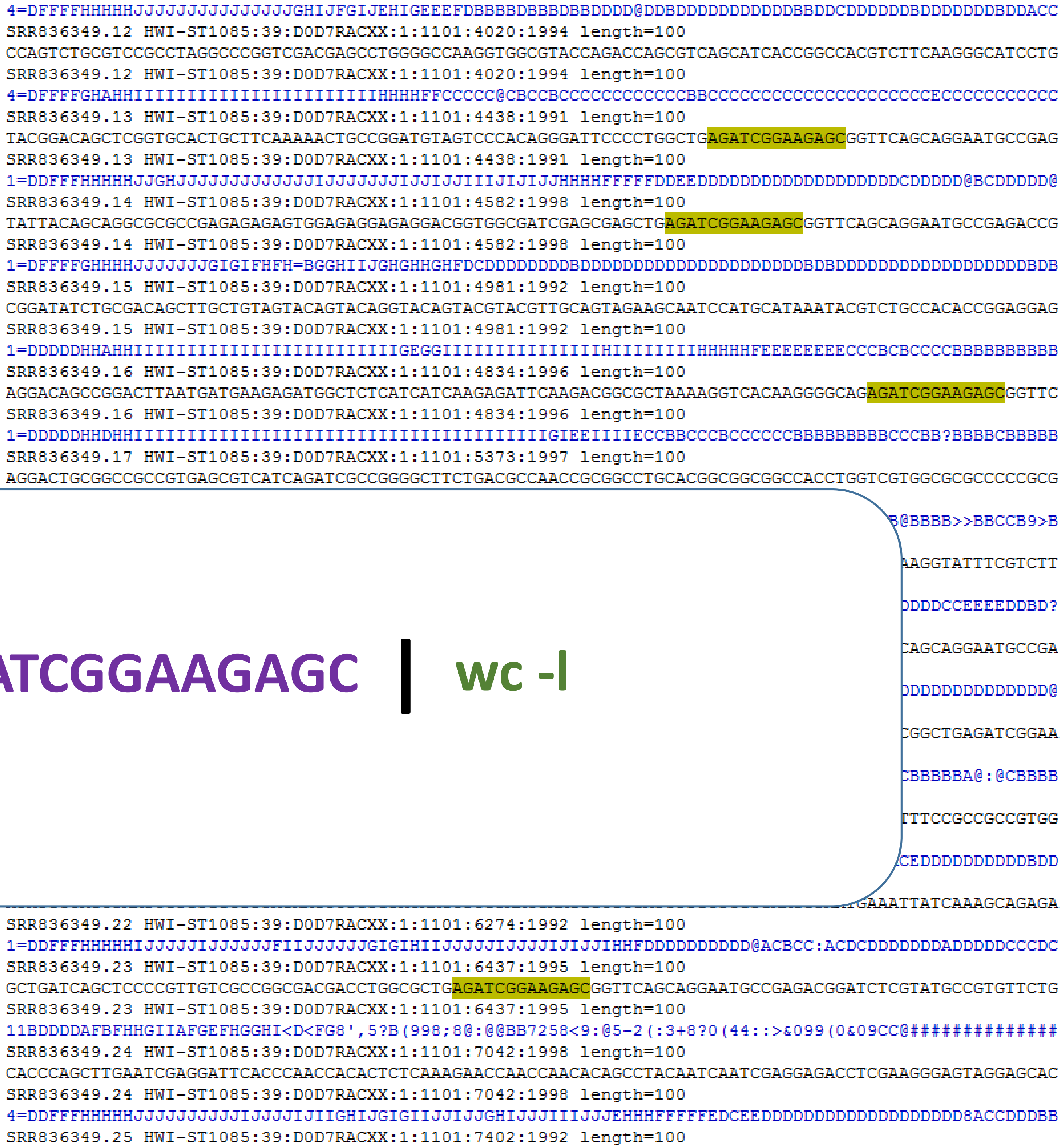
- **Count the selected lines**

  `wc -l`



```
4=DFFFFHHHHHJJJJJJJJJJJJJJJGHIJFGIJEHIGEEEFDBBBBDBBBDBBDDDD@DDBDDDDDDDDDDDBBDDCDDDDDDBDDDDDDDBDDACC
SRR836349.12 HWI-ST1085:39:D0D7RACXX:1:1101:4020:1994 length=100
CCAGTCTGCGTCCGCCTAGGCCCGGTCGACGAGCCTGGGGCCAAGGTGGCGTACCAGACCAGCGTCAGCATCACCGGCCACGTCTTCAAGGGCATCCTG
SRR836349.12 HWI-ST1085:39:D0D7RACXX:1:1101:4020:1994 length=100
4=DFFFFGHAHHIIIIIIIIIIIIIIIIIIIIIIIHHHHFFCCCCC@CBCCBCCCCCCCCCCCCBBCCCCCCCCCCCCCCCCCCCCCECCCCCCCCCCC
SRR836349.13 HWI-ST1085:39:D0D7RACXX:1:1101:4438:1991 length=100
TACGGACAGCTCGGTGCACTGCTTCAAAAACTGCCGGATGTAGTCCCACAGGGATTCCCCTGGCTGAGATCGGAAGAGCGGTTCAGCAGGAATGCCGAG
SRR836349.13 HWI-ST1085:39:D0D7RACXX:1:1101:4438:1991 length=100
1=DDFFFHHHHHJJGHJJJJJJJJJJJJJIJJJJJJJIJJIJJIIIJIJIJJHHHHFFFFFDDEEDDDDDDDDDDDDDDDDDDDCDDDDD@BCDDDDD@
SRR836349.14 HWI-ST1085:39:D0D7RACXX:1:1101:4582:1998 length=100
TATTACAGCAGGCGCGCCGAGAGAGAGTGGAGAGGAGAGGACGGTGGCGATCGAGCGAGCTGAGATCGGAAGAGCGGTTCAGCAGGAATGCCGAGACCG
SRR836349.14 HWI-ST1085:39:D0D7RACXX:1:1101:4582:1998 length=100
1=DFFFFGHHHHJJJJJJGIGIFHFH=BGGHIIJGHGHHGHFDCDDDDDDDDBDDDDDDDDDDDDDDDDDDDDDDBDBDDDDDDDDDDDDDDDDDDBDB
SRR836349.15 HWI-ST1085:39:D0D7RACXX:1:1101:4981:1992 length=100
CGGATATCTGCGACAGCTTGCTGTAGTACAGTACAGGTACAGTACGTACGTTGCAGTAGAAGCAATCCATGCATAAATACGTCTGCCACACCGGAGGAG
SRR836349.15 HWI-ST1085:39:D0D7RACXX:1:1101:4981:1992 length=100
1=DDDDDHHAHHIIIIIIIIIIIIIIIIIIIIIIIIIIGEGGIIIIIIIIIIIIIIIIIHIIIIIIIIIHHHHHFEEEEEEEECCCBCBCCCCBBBBBBBBBB
SRR836349.16 HWI-ST1085:39:D0D7RACXX:1:1101:4834:1996 length=100
AGGACAGCCGGACTTAATGATGAAGAGATGGCTCTCATCATCAAGAGATTCAAGACGGCGCTAAAAGGTCACAAGGGGCAGAGATCGGAAGAGCGGTTC
SRR836349.16 HWI-ST1085:39:D0D7RACXX:1:1101:4834:1996 length=100
1=DDDDDHHDHHIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIGIEEIIIIECCBBCCCBCCCCCCBBBBBBBBBCCCBB?BBBBCBBBBB
SRR836349.17 HWI-ST1085:39:D0D7RACXX:1:1101:5373:1997 length=100
AGGACTGCGGCCGCCGTGAGCGTCATCAGATCGCCGGGGCTTCTGACGCCAACCGCGGCCTGCACGGCGGCGGCCACCTGGTCGTGGCGCGCCCCCGCG
SRR836349.17 HWI-ST1085:39:D0D7RACXX:1:1101:5373:1997 length=100
1=DFFFFHHAHHIIIIIIIIIIIIIIIIIIIGIIIIHFFCCCCCCCECCCCBCBCCCCCCCCCCCCCCBBCCBBBB>5@BACCACB@BBBB>>BBCCB9>B
SRR836349.18 HWI-ST1085:39:D0D7RACXX:1:1101:5308:1997 length=100
TAGCCAGCGACCGCTACGTGGCAGTCGTCGTGAGGACACCTAATCCTCAATTCGCATAAATATTCACCTACACCCCTTGCGTCCAAGGTATTTCGTCTT
SRR836349.18 HWI-ST1085:39:D0D7RACXX:1:1101:5308:1997 length=100
4=DDFFFHHHHHJJJJJJIJJJJJJJJJJJJIHIGIIGGIJHGHHFFFFFFEEEEDDDDDDEEEFDEDDDDDACCBBDDDDDDDDDDDDDCCEEEEDDBD?
SRR836349.19 HWI-ST1085:39:D0D7RACXX:1:1101:5732:1994 length=100
ATCTCTGCTTGATTAGAAAAAGGAGATCCACGGCCTGTGAGTAACTACGCCATGTTACGAATCACAGAGATCGGAAGAGCGGTTCAGCAGGAATGCCGA
SRR836349.19 HWI-ST1085:39:D0D7RACXX:1:1101:5732:1994 length=100
1=DDFFFHHCACFHIJJJJJJJJJJJJJJJJJJJJJJJIJIIFHIIJIJJJJJJJJJJJJHHHGFFFFFEEEEDDDDDDDDDDDDDDDDDDDDDDDDDD@
SRR836349.20 HWI-ST1085:39:D0D7RACXX:1:1101:5623:1995 length=100
TCCTGGACTGCACCCTTAGGCGACTTCTGCGCGGATAGTCTTCGCGATAGACATTGGATTTTCTCGCACTTGTTGTGCTTAGTCCGGCTGAGATCGGAA
SRR836349.20 HWI-ST1085:39:D0D7RACXX:1:1101:5623:1995 length=100
4=DDDDDHHBHHIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIHEECCCCCCCFFCCCCCCECCBBBBBCCCCBCBCCCCCCCCCBBBBBA@:@CBBBB
SRR836349.21 HWI-ST1085:39:D0D7RACXX:1:1101:6484:1991 length=100
GAAGACTGCGACTTACCGTGTTGGAGTTGTGTAGCCACCGTGAAACACCACACTGCGCGCCGCCGCCACCACCAAAAGAGAGTGTTTCCGCCGCCGTGG
SRR836349.21 HWI-ST1085:39:D0D7RACXX:1:1101:6484:1991 length=100
4=DDFFFGHHGHJJIJJJJJJJJJJIJJIJJJJJIJJJJJJJJJJJHIIJJJIJJIJHHFDDDDDDDDDDDDDDDDDDDDDDD:ACEDDDDDDDDDDBDD
SRR836349.22 HWI-ST1085:39:D0D7RACXX:1:1101:6274:1992 length=100
ATATGCCAGCTAATCGTGTGGCTTTAGAAGCCTGTGTACAAGCTCGTAACGAAGGGCGCGATCTTGCTCGTGAAGGTAATGAAATTATCAAAGCAGAGA
SRR836349.22 HWI-ST1085:39:D0D7RACXX:1:1101:6274:1992 length=100
1=DDFFFHHHHHIJJJJJIJJJJJJFIIJJJJJJGIGIHIIJJJJJIJJJJIJIJJIHHFDDDDDDDDDD@ACBCC:ACDCDDDDDDDADDDDDCCCDC
SRR836349.23 HWI-ST1085:39:D0D7RACXX:1:1101:6437:1995 length=100
GCTGATCAGCTCCCCGTTGTCGCCGGCGACGACCTGGCGCTGAGATCGGAAGAGCGGTTCAGCAGGAATGCCGAGACGGATCTCGTATGCCGTGTTCTG
SRR836349.23 HWI-ST1085:39:D0D7RACXX:1:1101:6437:1995 length=100
11BDDDDAFBFHHGIIAFGEFHGGHI<D<FG8',5?B(998;8@:@@BB7258<9:@5-2(:3+8?0(44::>&099(0&09CC@##############
SRR836349.24 HWI-ST1085:39:D0D7RACXX:1:1101:7042:1998 length=100
CACCCAGCTTGAATCGAGGATTCACCCAACCACACTCTCAAAGAACCAACCAACACAGCCTACAATCAATCGAGGAGACCTCGAAGGGAGTAGGAGCAC
SRR836349.24 HWI-ST1085:39:D0D7RACXX:1:1101:7042:1998 length=100
4=DDFFFHHHHHJJJJJJJJJJJIJJJJIJIIGHIJGIGIIJJIJJGHIJJJIIIJJJEHHHFFFFFEDCEEDDDDDDDDDDDDDDDDDDD8ACCDDDBB
SRR836349.25 HWI-ST1085:39:D0D7RACXX:1:1101:7402:1992 length=100
```

Estimate the percentage of sequencing reads in the FASTQ file that contains the adapter “AGATCGGAAGAGC”.



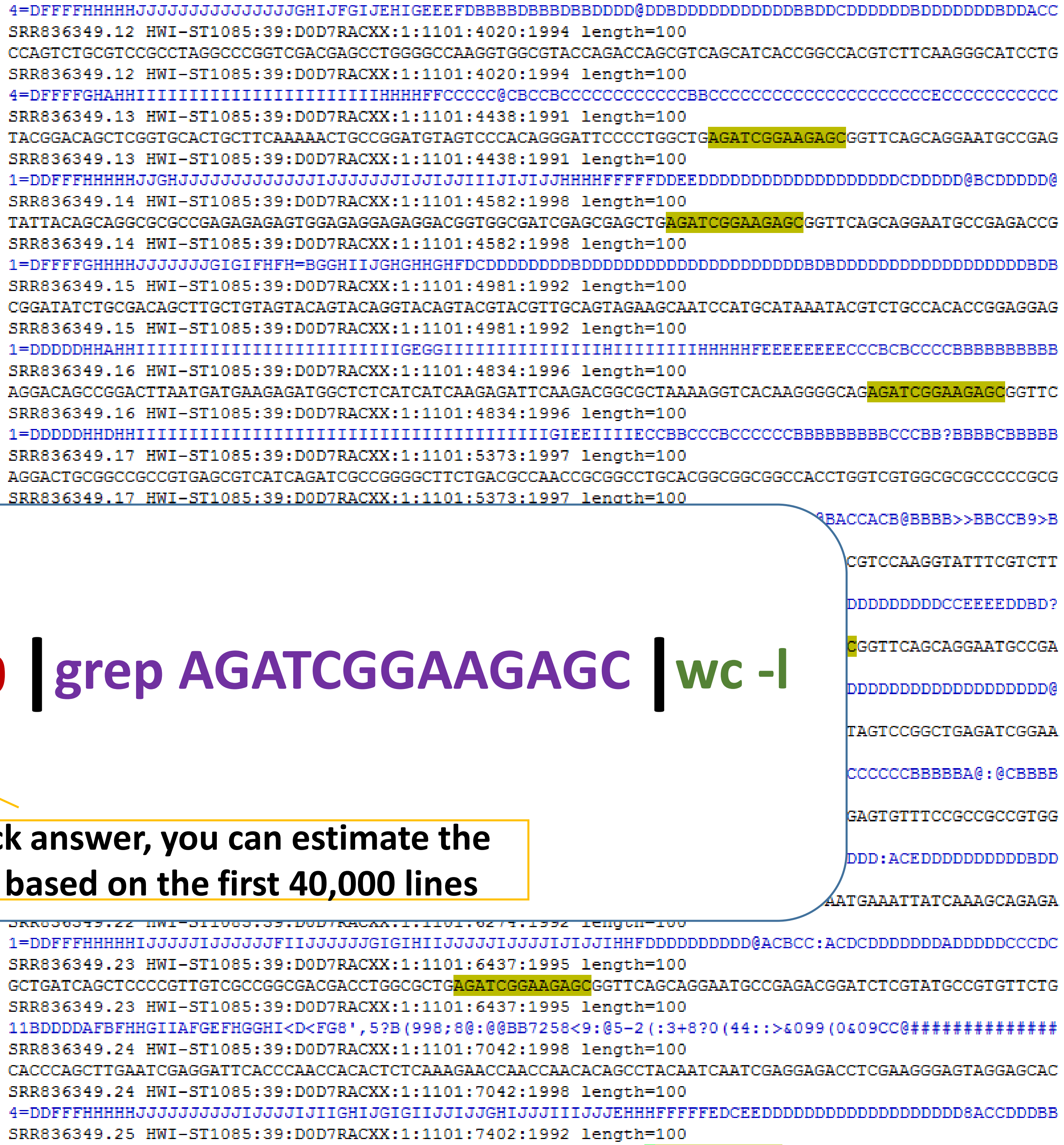
```
4=DFFFFHHHHHJJJJJJJJJJJJJJJGHIJFGIJEHIGEEEFDBBBBDBBBDBBDDDD@DDBDDDDDDDDDDDDBBDDCDDDDDDBDDDDDDDDBDDACC
SRR836349.12 HWI-ST1085:39:D0D7RACXX:1:1101:4020:1994 length=100
CCAGTCTGCGTCCGCCTAGGCCCGGTCGACGAGCCTGGGGCCAAGGTGGCGTACCAGACCAGCGTCAGCATCACCGGCCACGTCTTCAAGGGCATCCTG
SRR836349.12 HWI-ST1085:39:D0D7RACXX:1:1101:4020:1994 length=100
4=DFFFFGHAHHIIIIIIIIIIIIIIIIIIIIIIHHHHFFCCCCC@CBCCBCCCCCCCCCCCCBBCCCCCCCCCCCCCCCCCCCCCECCCCCCCCCCC
SRR836349.13 HWI-ST1085:39:D0D7RACXX:1:1101:4438:1991 length=100
TACGGACAGCTCGGTGCACTGCTTCAAAAACTGCCGGATGTAGTCCCACAGGGATTCCCCTGGCTGAGATCGGAAGAGCGGTTCAGCAGGAATGCCGAG
SRR836349.13 HWI-ST1085:39:D0D7RACXX:1:1101:4438:1991 length=100
1=DDFFFHHHHHJJGHJJJJJJJJJJJJJIJJJJJJJIJJIJJIIIJIJIJJHHHHFFFFFDDEEDDDDDDDDDDDDDDDDDDDCDDDDD@BCDDDDD@
SRR836349.14 HWI-ST1085:39:D0D7RACXX:1:1101:4582:1998 length=100
TATTACAGCAGGCGCGCCGAGAGAGAGTGGAGAGGAGAGGACGGTGGCGATCGAGCGAGCTGAGATCGGAAGAGCGGTTCAGCAGGAATGCCGAGACCG
SRR836349.14 HWI-ST1085:39:D0D7RACXX:1:1101:4582:1998 length=100
1=DFFFFGHHHHJJJJJJGIGIFHFH=BGGHIIJGHGHHGHFDCDDDDDDDDBDDDDDDDDDDDDDDDDDDDDDBDBDDDDDDDDDDDDDDDDDDDBDB
SRR836349.15 HWI-ST1085:39:D0D7RACXX:1:1101:4981:1992 length=100
CGGATATCTGCGACAGCTTGCTGTAGTACAGTACAGGTACAGTACGTACGTTGCAGTAGAAGCAATCCATGCATAAATACGTCTGCCACACCGGAGGAG
SRR836349.15 HWI-ST1085:39:D0D7RACXX:1:1101:4981:1992 length=100
1=DDDDDHHAHHIIIIIIIIIIIIIIIIIIIIIIIIGEGGIIIIIIIIIIIIIIIIHIIIIIIIIHHHHHFEEEEEEEECCCBCBCCCCBBBBBBBBBB
SRR836349.16 HWI-ST1085:39:D0D7RACXX:1:1101:4834:1996 length=100
AGGACAGCCGGACTTAATGATGAAGAGATGGCTCTCATCATCAAGAGATTCAAGACGGCGCTAAAAGGTCACAAGGGGCAGAGATCGGAAGAGCGGTTC
SRR836349.16 HWI-ST1085:39:D0D7RACXX:1:1101:4834:1996 length=100
1=DDDDDHHDHHIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIGIEEIIIIECCBBCCCBCCCCCCBBBBBBBBBCCCBB?BBBBCBBBBB
SRR836349.17 HWI-ST1085:39:D0D7RACXX:1:1101:5373:1997 length=100
AGGACTGCGGCCGCCGTGAGCGTCATCAGATCGCCGGGGCTTCTGACGCCAACCGCGGCCTGCACGGCGGCGGCCACCTGGTCGTGGCGCGCCCCCGCG
B@BBBB>>BBCCB9>B
AAGGTATTTCGTCTT
DDDDDCCEEEEEDDBD?
CAGCAGGAATGCCGA
DDDDDDDDDDDDDD@
CGGCTGAGATCGGAA
CBBBBBA@:@CBBBB
TTTCCGCCGCCGTGG
CEDDDDDDDDDDBDD
GAAATTATCAAAGCAGAGA
SRR836349.22 HWI-ST1085:39:D0D7RACXX:1:1101:6274:1992 length=100
1=DDFFFHHHHHIJJJJJIJJJJJJFIIJJJJJJGIGIHIIJJJJJIJJJJIJIJJIHHFDDDDDDDDDD@ACBCC:ACDCDDDDDDDADDDDDCCCDC
SRR836349.23 HWI-ST1085:39:D0D7RACXX:1:1101:6437:1995 length=100
GCTGATCAGCTCCCCGTTGTCGCCGGCGACGACCTGGCGCTGAGATCGGAAGAGCGGTTCAGCAGGAATGCCGAGACGGATCTCGTATGCCGTGTTCTG
SRR836349.23 HWI-ST1085:39:D0D7RACXX:1:1101:6437:1995 length=100
11BDDDDAFBFHHGIIAFGEFHGGHI<D<FG8',5?B(998;8@:@@BB7258<9:@5-2(:3+8?0(44::>&099(0&09CC@##############
SRR836349.24 HWI-ST1085:39:D0D7RACXX:1:1101:7042:1998 length=100
CACCCAGCTTGAATCGAGGATTCACCCAACCACACTCTCAAAGAACCAACCAACACAGCCTACAATCAATCGAGGAGACCTCGAAGGGAGTAGGAGCAC
SRR836349.24 HWI-ST1085:39:D0D7RACXX:1:1101:7042:1998 length=100
4=DDFFFHHHHHJJJJJJJJJJIJJJJIJIIGHIJGIGIIJJIJJGHIJJJIIIJJJEHHHFFFFFEDCEEDDDDDDDDDDDDDDDDDD8ACCDDDBB
SRR836349.25 HWI-ST1085:39:D0D7RACXX:1:1101:7402:1992 length=100
```

```
cat  D7RACXX.fastq  | grep AGATCGGAAGAGC | wc -l
```

Estimate the percentage of sequencing reads in the FASTQ file that contains the adapter “AGATCGGAAGAGC”.

```
cat DeD7RACXX.fastq | head -n 40000 |grep AGATCGGAAGAGC |wc -l
```

To get a quick answer, you can estimate the percentage based on the first 40,000 lines

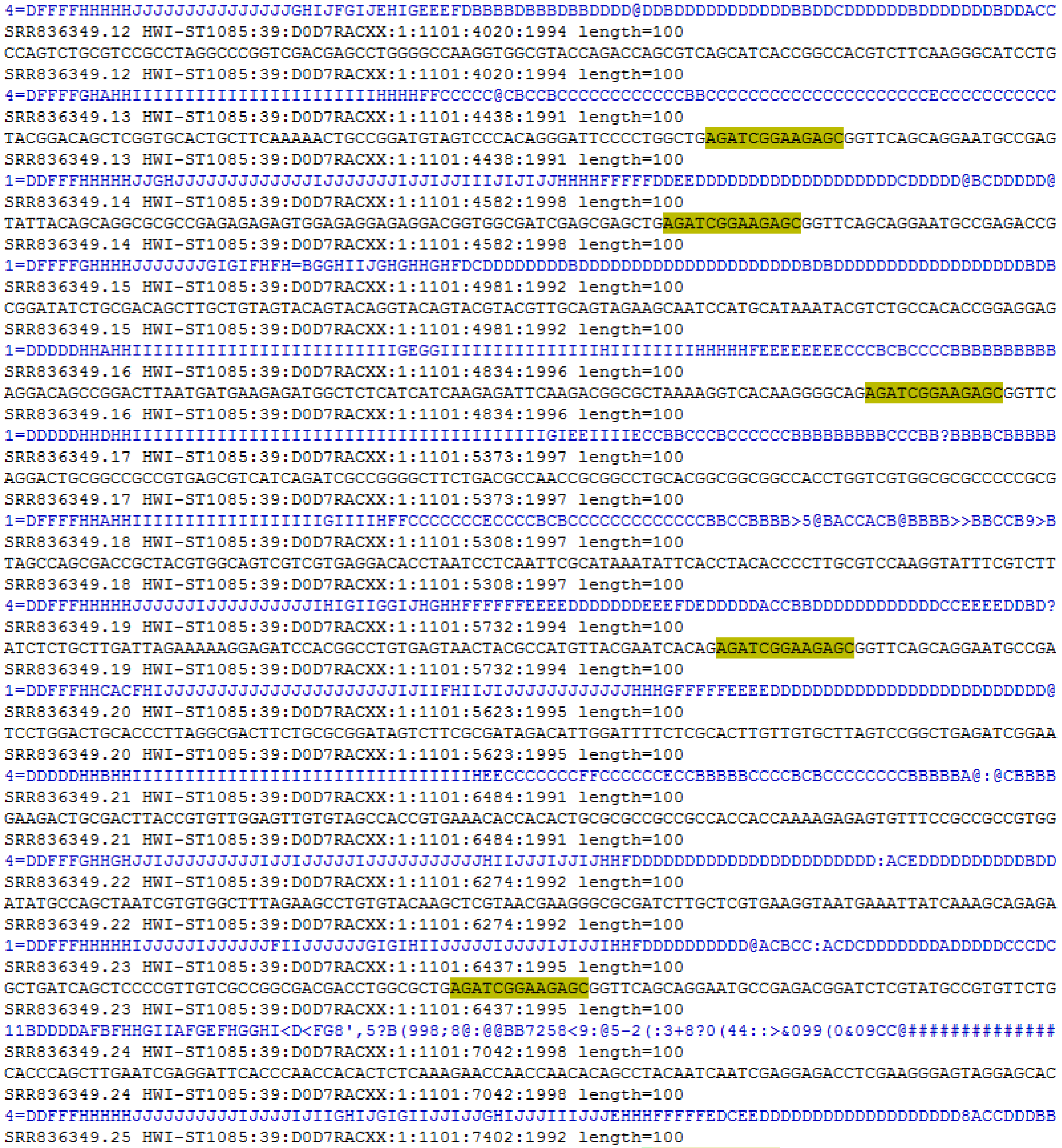


```
4=DFFFFHHHHHJJJJJJJJJJJJJJJJGHIJFGIJEHIGEEEFDBBBBDBBBDBBDDDD@DDBDDDDDDDDDDDDBBDDCDDDDDDBDDDDDDDBDDACC
SRR836349.12 HWI-ST1085:39:D0D7RACXX:1:1101:4020:1994 length=100
CCAGTCTGCGTCCGCCTAGGCCCGGTCGACGAGCCTGGGGCCAAGGTGGCGTACCAGACCAGCGTCAGCATCACCGGCCACGTCTTCAAGGGCATCCTG
SRR836349.12 HWI-ST1085:39:D0D7RACXX:1:1101:4020:1994 length=100
4=DFFFFGHAHHIIIIIIIIIIIIIIIIIIIIIIIIIHHHHFFCCCCC@CBCCBCCCCCCCCCCCCBBCCCCCCCCCCCCCCCCCCCCCECCCCCCCCCCC
SRR836349.13 HWI-ST1085:39:D0D7RACXX:1:1101:4438:1991 length=100
TACGGACAGCTCGGTGCACTGCTTCAAAAACTGCCGGATGTAGTCCCACAGGGATTCCCCTGGCTGAGATCGGAAGAGCGGTTCAGCAGGAATGCCGAG
SRR836349.13 HWI-ST1085:39:D0D7RACXX:1:1101:4438:1991 length=100
1=DDFFFHHHHHJJGHJJJJJJJJJJJJJIJJJJJJJIJJIJJIIIJIJIJJHHHHFFFFFDDEEDDDDDDDDDDDDDDDDDDDDCDDDDD@BCDDDDD@
SRR836349.14 HWI-ST1085:39:D0D7RACXX:1:1101:4582:1998 length=100
TATTACAGCAGGCGCGCCGAGAGAGAGTGGAGAGGAGAGGACGGTGGCGATCGAGCGAGCTGAGATCGGAAGAGCGGTTCAGCAGGAATGCCGAGACCG
SRR836349.14 HWI-ST1085:39:D0D7RACXX:1:1101:4582:1998 length=100
1=DFFFFGHHHHJJJJJJGIGIFHFH=BGGHIIJGHGHHGHFDCDDDDDDDDBDDDDDDDDDDDDDDDDDDDDDDBDBDDDDDDDDDDDDDDDDDDDBDB
SRR836349.15 HWI-ST1085:39:D0D7RACXX:1:1101:4981:1992 length=100
CGGATATCTGCGACAGCTTGCTGTAGTACAGTACAGGTACAGTACGTACGTTGCAGTAGAAGCAATCCATGCATAAATACGTCTGCCACACCGGAGGAG
SRR836349.15 HWI-ST1085:39:D0D7RACXX:1:1101:4981:1992 length=100
1=DDDDDHHAHHIIIIIIIIIIIIIIIIIIIIIIIIGEGGIIIIIIIIIIIIIIIHIIIIIIIIIHHHHHFEEEEEEEECCCBCBCCCCBBBBBBBBBB
SRR836349.16 HWI-ST1085:39:D0D7RACXX:1:1101:4834:1996 length=100
AGGACAGCCGGACTTAATGATGAAGAGATGGCTCTCATCATCAAGAGATTCAAGACGGCGCTAAAAGGTCACAAGGGGCAGAGATCGGAAGAGCGGTTC
SRR836349.16 HWI-ST1085:39:D0D7RACXX:1:1101:4834:1996 length=100
1=DDDDDHHDHHIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIGIEEIIIIECCBBCCCBCCCCCCBBBBBBBBBCCCBB?BBBBCBBBBB
SRR836349.17 HWI-ST1085:39:D0D7RACXX:1:1101:5373:1997 length=100
AGGACTGCGGCCGCCGTGAGCGTCATCAGATCGCCGGGGCTTCTGACGCCAACCGCGGCCTGCACGGCGGCGGCCACCTGGTCGTGGCGCGCCCCCGCG
SRR836349.17 HWI-ST1085:39:D0D7RACXX:1:1101:5373:1997 length=100
BACCACB@BBBB>>BBCCB9>B
CGTCCAAGGTATTTCGTCTT
DDDDDDDDDCCEEEEDDBD?
CGGTTCAGCAGGAATGCCGA
DDDDDDDDDDDDDDDDDDDD@
TAGTCCGGCTGAGATCGGAA
CCCCCCBBBBBA@:@CBBBB
GAGTGTTTCCGCCGCCGTGG
DDD:ACEDDDDDDDDDDBDD
AATGAAATTATCAAAGCAGAGA
SRR836349.22 HWI-ST1085:39:D0D7RACXX:1:1101:6274:1992 length=100
1=DDFFFHHHHHIJJJJIJJJJJJFIIJJJJJJGIGIHIIJJJJJIJJJJIJIJJIHHFDDDDDDDDDD@ACBCC:ACDCDDDDDDDADDDDDCCCDC
SRR836349.23 HWI-ST1085:39:D0D7RACXX:1:1101:6437:1995 length=100
GCTGATCAGCTCCCCGTTGTCGCCGGCGACGACCTGGCGCTGAGATCGGAAGAGCGGTTCAGCAGGAATGCCGAGACGGATCTCGTATGCCGTGTTCTG
SRR836349.23 HWI-ST1085:39:D0D7RACXX:1:1101:6437:1995 length=100
11BDDDDAFBFHHGIIAFGEFHGGHI<D<FG8',5?B(998;8@:@@BB7258<9:@5-2(:3+8?0(44::>&099(0&09CC@##############
SRR836349.24 HWI-ST1085:39:D0D7RACXX:1:1101:7042:1998 length=100
CACCCAGCTTGAATCGAGGATTCACCCAACCACACTCTCAAAGAACCAACCAACACAGCCTACAATCAATCGAGGAGACCTCGAAGGGAGTAGGAGCAC
SRR836349.24 HWI-ST1085:39:D0D7RACXX:1:1101:7042:1998 length=100
4=DDFFFHHHHHJJJJJJJJJJIJJJJIJIIGHIJGIGIIJJIJJGHIJJJIIIJJJEHHHFFFFFEDCEEDDDDDDDDDDDDDDDDDDDD8ACCDDDBB
SRR836349.25 HWI-ST1085:39:D0D7RACXX:1:1101:7402:1992 length=100
```

Estimate the percentage of sequencing reads in the FASTQ file that contains the adapter "AGATCGGAAGAGC".

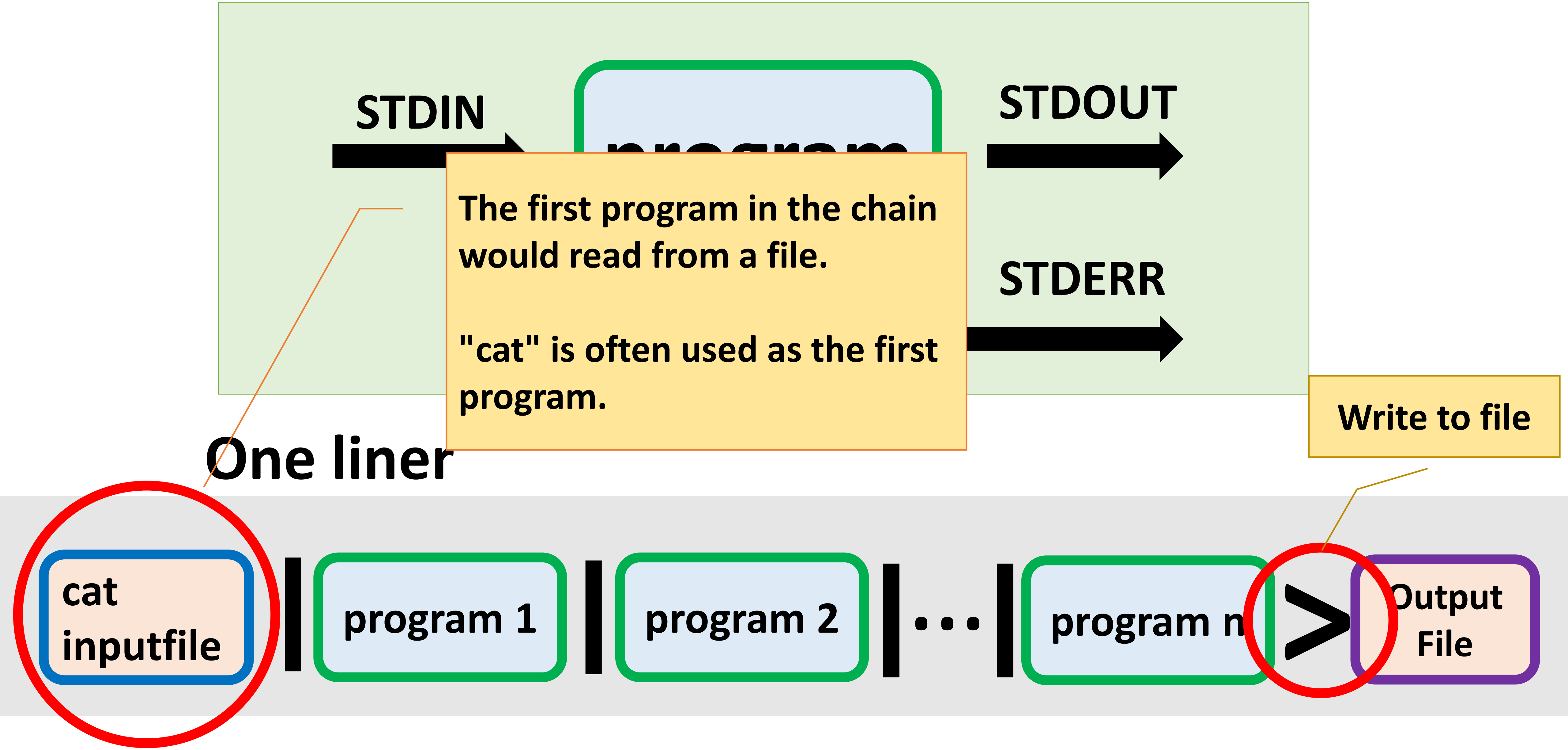
```
cat D7RACXX.fastq | \
head -n 40000 | \
grep AGATCGGAAGAGC | \
wc -l
```

Use "\" to separate the command into multiple lines



```
4=DFFFFHHHHHJJJJJJJJJJJJJJJGHIJFGIJEHIGEEEFDBBBBDBBBDBBDDDD@DDBDDDDDDDDDDDDBBDDCDDDDDDBDDDDDDDBDDACC
SRR836349.12 HWI-ST1085:39:D0D7RACXX:1:1101:4020:1994 length=100
CCAGTCTGCGTCCGCCTAGGCCCGGTCGACGAGCCTGGGGCCAAGGTGGCGTACCAGACCAGCGTCAGCATCACCGGCCACGTCTTCAAGGGCATCCTG
SRR836349.12 HWI-ST1085:39:D0D7RACXX:1:1101:4020:1994 length=100
4=DFFFFGHAHHIIIIIIIIIIIIIIIIIIIIIIHHHHFFCCCCC@CBCCBCCCCCCCCCCCCBBCCCCCCCCCCCCCCCCCCCCECCCCCCCCCCC
SRR836349.13 HWI-ST1085:39:D0D7RACXX:1:1101:4438:1991 length=100
TACGGACAGCTCGGTGCACTGCTTCAAAAACTGCCGGATGTAGTCCCACAGGGATTCCCCTGGCTGAGATCGGAAGAGCGGTTCAGCAGGAATGCCGAG
SRR836349.13 HWI-ST1085:39:D0D7RACXX:1:1101:4438:1991 length=100
1=DDFFFHHHHHJJGHJJJJJJJJJJJJJIJJJJJJJIJJIJJIIIJIJIJJHHHHFFFFFDDEEDDDDDDDDDDDDDDDDDDDCDDDDD@BCDDDDD@
SRR836349.14 HWI-ST1085:39:D0D7RACXX:1:1101:4582:1998 length=100
TATTACAGCAGGCGCGCCGAGAGAGAGTGGAGAGGAGAGGACGGTGGCGATCGAGCGAGCTGAGATCGGAAGAGCGGTTCAGCAGGAATGCCGAGACCG
SRR836349.14 HWI-ST1085:39:D0D7RACXX:1:1101:4582:1998 length=100
1=DFFFFGHHHHJJJJJJGIGIFHFH=BGGHIIJGHGHHGHFDCDDDDDDDDBDDDDDDDDDDDDDDDDDDDDDBDBDDDDDDDDDDDDDDDDDDBDB
SRR836349.15 HWI-ST1085:39:D0D7RACXX:1:1101:4981:1992 length=100
CGGATATCTGCGACAGCTTGCTGTAGTACAGTACAGGTACAGTACGTACGTTGCAGTAGAAGCAATCCATGCATAAATACGTCTGCCACACCGGAGGAG
SRR836349.15 HWI-ST1085:39:D0D7RACXX:1:1101:4981:1992 length=100
1=DDDDDHHAHHIIIIIIIIIIIIIIIIIIIIIIIIIGEGGIIIIIIIIIIIIIIIIHIIIIIIIIHHHHHFEEEEEEEECCCBCBCCCCBBBBBBBBBB
SRR836349.16 HWI-ST1085:39:D0D7RACXX:1:1101:4834:1996 length=100
AGGACAGCCGGACTTAATGATGAAGAGATGGCTCTCATCATCAAGAGATTCAAGACGGCGCTAAAAGGTCACAAGGGGCAGAGATCGGAAGAGCGGTTC
SRR836349.16 HWI-ST1085:39:D0D7RACXX:1:1101:4834:1996 length=100
1=DDDDDHHDHHIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIGIEEIIIIECCBBCCCBCCCCCCBBBBBBBBCCCBB?BBBBCBBBBB
SRR836349.17 HWI-ST1085:39:D0D7RACXX:1:1101:5373:1997 length=100
AGGACTGCGGCCGCCGTGAGCGTCATCAGATCGCCGGGGCTTCTGACGCCAACCGCGGCCTGCACGGCGGCGGCCACCTGGTCGTGGCGCGCCCCCGCG
SRR836349.17 HWI-ST1085:39:D0D7RACXX:1:1101:5373:1997 length=100
1=DFFFFHHAHHIIIIIIIIIIIIIIIIIIGIIIIHFFCCCCCCCECCCCBCBCCCCCCCCCCCCCBBCCBBBB>5@BACCACB@BBBB>>BBCCB9>B
SRR836349.18 HWI-ST1085:39:D0D7RACXX:1:1101:5308:1997 length=100
TAGCCAGCGACCGCTACGTGGCAGTCGTCGTGAGGACACCTAATCCTCAATTCGCATAAATATTCACCTACACCCCTTGCGTCCAAGGTATTTCGTCTT
SRR836349.18 HWI-ST1085:39:D0D7RACXX:1:1101:5308:1997 length=100
4=DDFFFHHHHHJJJJJJIJJJJJJJJJJJIHIGIIGGIJHGHHFFFFFFEEEEDDDDDDEEEFDEDDDDDACCBBDDDDDDDDDDDDDCCEEEEDDBD?
SRR836349.19 HWI-ST1085:39:D0D7RACXX:1:1101:5732:1994 length=100
ATCTCTGCTTGATTAGAAAAAGGAGATCCACGGCCTGTGAGTAACTACGCCATGTTACGAATCACAGAGATCGGAAGAGCGGTTCAGCAGGAATGCCGA
SRR836349.19 HWI-ST1085:39:D0D7RACXX:1:1101:5732:1994 length=100
1=DDFFFHHCACFHIJJJJJJJJJJJJJJJJJJJJJJIJIIFHIIJIJJJJJJJJJJJHHHGFFFFFEEEEDDDDDDDDDDDDDDDDDDDDDDDDDD@
SRR836349.20 HWI-ST1085:39:D0D7RACXX:1:1101:5623:1995 length=100
TCCTGGACTGCACCCTTAGGCGACTTCTGCGCGGATAGTCTTCGCGATAGACATTGGATTTTCTCGCACTTGTTGTGCTTAGTCCGGCTGAGATCGGAA
SRR836349.20 HWI-ST1085:39:D0D7RACXX:1:1101:5623:1995 length=100
4=DDDDDHHBHHIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIHEECCCCCCCFFCCCCCCECCBBBBBCCCCBCBCCCCCCCCCBBBBBA@:@CBBBB
SRR836349.21 HWI-ST1085:39:D0D7RACXX:1:1101:6484:1991 length=100
GAAGACTGCGACTTACCGTGTTGGAGTTGTGTAGCCACCGTGAAACACCACACTGCGCGCCGCCGCCACCACCAAAAGAGAGTGTTTCCGCCGCCGTGG
SRR836349.21 HWI-ST1085:39:D0D7RACXX:1:1101:6484:1991 length=100
4=DDFFFGHHGHJJIJJJJJJJJJIJJIJJJJJIJJJJJJJJJJJHIIJJJIJJIJHHFDDDDDDDDDDDDDDDDDDDDDDDD:ACEDDDDDDDDDDBDD
SRR836349.22 HWI-ST1085:39:D0D7RACXX:1:1101:6274:1992 length=100
ATATGCCAGCTAATCGTGTGGCTTTAGAAGCCTGTGTACAAGCTCGTAACGAAGGGCGCGATCTTGCTCGTGAAGGTAATGAAATTATCAAAGCAGAGA
SRR836349.22 HWI-ST1085:39:D0D7RACXX:1:1101:6274:1992 length=100
1=DDFFFHHHHHIJJJJJIJJJJJJFIIJJJJJJGIGIHIIJJJJJIJJJJIJIJJIHHFDDDDDDDDDD@ACBCC:ACDCDDDDDDDADDDDDCCCDC
SRR836349.23 HWI-ST1085:39:D0D7RACXX:1:1101:6437:1995 length=100
GCTGATCAGCTCCCCGTTGTCGCCGGCGACGACCTGGCGCTGAGATCGGAAGAGCGGTTCAGCAGGAATGCCGAGACGGATCTCGTATGCCGTGTTCTG
SRR836349.23 HWI-ST1085:39:D0D7RACXX:1:1101:6437:1995 length=100
11BDDDDAFBFHHGIIAFGEFHGGHI<D<FG8',5?B(998;8@:@@BB7258<9:@5-2(:3+8?0(44::>&099(0&09CC@##############
SRR836349.24 HWI-ST1085:39:D0D7RACXX:1:1101:7042:1998 length=100
CACCCAGCTTGAATCGAGGATTCACCCAACCACACTCTCAAAGAACCAACCAACACAGCCTACAATCAATCGAGGAGACCTCGAAGGGAGTAGGAGCAC
SRR836349.24 HWI-ST1085:39:D0D7RACXX:1:1101:7042:1998 length=100
4=DDFFFHHHHHJJJJJJJJJJIJJJJIJIIGHIJGIGIIJJIJJGHIJJJIIIJJJEHHHFFFFFEDCEEDDDDDDDDDDDDDDDDDD8ACCDDDBB
SRR836349.25 HWI-ST1085:39:D0D7RACXX:1:1101:7402:1992 length=100
```

Three streams for a standard Linux program



STDIN

program

STDOUT

STDERR

The first program in the chain would read from a file.

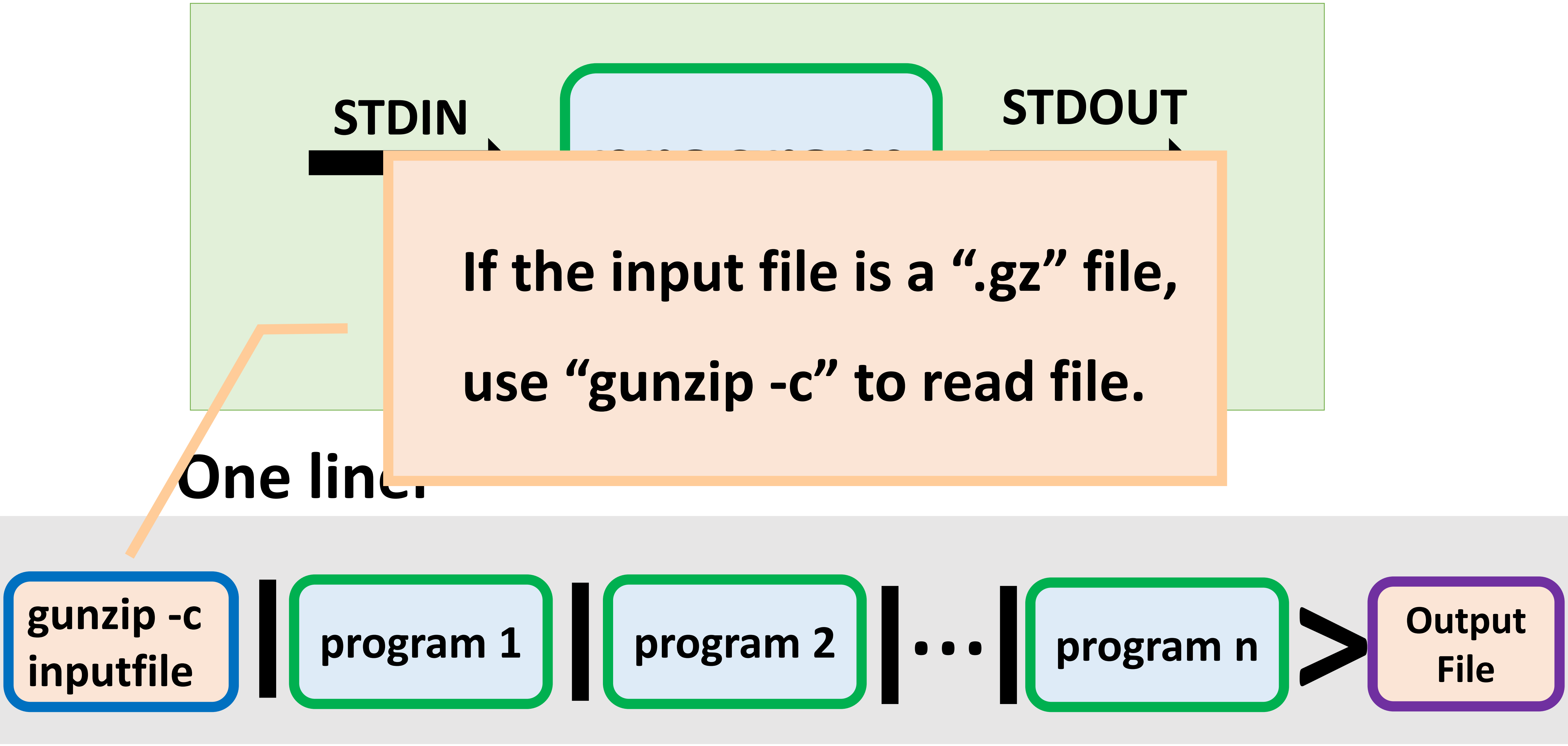
"cat" is often used as the first program.

One liner

cat inputfile | program 1 | program 2 |···| program n > Output File

Write to file

Three streams for a standard Linux program



STDIN

STDOUT

If the input file is a ".gz" file,

use "gunzip -c" to read file.

One lin

gunzip -c inputfile | program 1 | program 2 | ··· | program n > Output File

grep

Search for a pattern and output matched lines

```
$ cat mydata.txt

AAGATCAAAAAAGA

ATTTACGAAAAAAGA

ACCTGTTGGATCCAAAGTT

AAACTTTCGACGATCT

ATTTTTTTAGAAAGG
```

```
$ cat mydata.txt | grep '[AC]GATC'

AAGATCAAAAAAGA

AAACTTTCGACGATCT
```

wc -l Count the number of lines

```
$ cat mydata.txt
AAGATCAAAAAAGA
ATTTACGAAAAAAGA
ACCTGTTGGATCCAAAGTT
AAACTTTCGACGATCT
ATTTTTTTAGAAAGG
```

```
$ cat mydata.txt | grep '[AC]GATC' | wc -l
2
```

sort Sort the text in a file

```
$ sort myChr.txt
```

```
Chr1
Chr10
Chr2
Chr3
Chr4
Chr5
```

```
$ sort -V myChr.txt
```

```
Chr1
Chr2
Chr3
Chr4
Chr5
Chr10
```

```
$ sort -n myPos.txt
```

```
1
2
3
4
5
10
```

sort

Sort the text by multiple columns

```
$ sort -k1,1V -k2,2n myChrPos.txt
```

```
Chr1      12
Chr1      100
Chr1      200
Chr2      34
Chr2      121
Chr2      300
```

Locale and sorting

Computer English

`LC_ALL=C`

Alphabetic order
A
B
C
…
X
Y
Z
a
b
c
…
x
y
z

US English *

`LC_ALL=US_en`

Alphabetic order
a
A
b
B
c
C
…
x
X
y
Y
z
Z

* On Linux, US English locale sorting also ignore the non-alphanumerical characters. This could cause problems.

Use this Linux command to find out the locale setting on your server:

```
locale
```

Some extra parameter to set for the “sort” command

```
LC_ALL=C sort -S 4G -k1,1 myChr.txt
```

LC_ALL=C	**vs**	**LC_ALL=US_en**
AA		aa
BB		a.a
a.a		AA
aa		bb
bb		BB

* BioHPC default locale is C. So you can skip this parameter

-S 4G

Use RAM as buffer 4G

uniq -c

Count the occurrence of unique tags

```
$ cat mydata.txt
ItemB
ItemA
ItemB
ItemC
ItemB
ItemC
ItemB
ItemC
```

```
$ cat mydata.txt | sort | uniq -c

1       ItemA

4       ItemB

3       ItemC
```

Mark sure to run “sort” before “uniq”

Merging files:

cat f1 f2 vs **paste** f1 f2 vs **join** f1 f2

File 1:

Item1

Item2

File2:

Item3

Item4

```
cat File1 File2 > mergedfile1
```

```
Item1
Item2
Item3
Item4
```

```
paste File1 File2 > mergedfile2
```

```
Item1     Item3
Item2     Item4
```

* Make sure that that two files has same number of rows and sorted the same way. Otherwise, use “join”

join

Merging two files that share a common field

<u>File 1:</u>

Gene1	DNA-binding
Gene2	kinase
Gene3	membrane

<u>File2:</u>

Gene2	764
Gene3	23
Gene4	34

```
join  -1  1 -2  1  File1 File2 > mergedfile
```

```
Gene2         Kinase        764
Gene3         membrane   23
```

```
join  -1 1 -2 1  -a1 File1 File2 > mergedfile
```

```
Gene1         DNA-binding
Gene2         Kinase                  764
Gene3         membrane                23
```

cut

Output selected columns in a table

```
$ cat mydata.txt
```

```
Chr1 1000 2250 Gene1
Chr1 3010 5340 Gene2
Chr1 7500 8460 Gene3
Chr2 8933 9500 Gene4
```

```
$ cat mydata.txt | cut -f 1,4
```

```
Chr1 Gene1
Chr1 Gene2
Chr1 Gene3
Chr2 Gene4
```

sed

Modify text in a file

```
$ cat mydata.txt
```

```
Chr1 1000 2250 Gene1
Chr1 3010 5340 Gene2
Chr1 7500 8460 Gene3
Chr2 8933 9500 Gene4
```

```
$ cat mydata.txt | sed "s/^Chr//"
```

```
1       1000 2250 Gene1
1       3010 5340 Gene2
1       7500 8460 Gene3
2       8933 9500 Gene4
```

awk

Probably the most versatile function in Linux

```
$ cat mydata.txt
```

```
Chr1 1000 2250 Gene1
Chr1 3010 5340 Gene2
Chr1 7500 8460 Gene3
Chr2 8933 9500 Gene4
```

```
$ cat mydata.txt |\
awk '{if ($1=="Chr1") print $4}'
```

```
Gene1
Gene2
Gene3
```

A Good Practice:
Create a shell script file for the one liner

```
cat D7RACXX.fastq | \
head -n 40000 | \
grep AGATCGGAAGAGC | \
wc -l
```

Run the shell script

```
sh checkadapter.sh
```

Debug a one-liner

```
gunzip -c human.gff3.gz | \
awk 'BEGIN {OFS = "\t"}; {if ($3=="gene") print $1,$4-1,$5}' | \
bedtools coverage -a win1mb.bed -b stdin -counts | \
LC_ALL=C sort -k1,1V -k2,2n  > gene.cover.bed
```

```
gunzip -c human.gff3.gz | head -n 1000 > tmpfile
```

```
cat tmpfile| \
awk 'BEGIN {OFS = "\t"}; {if ($3=="gene") print $1,$4-1,$5}' | head -n 100
```

Many bioinformatics software support STDIN instead of input file

Run “BWA” without pipe:

```
bwa mem ref.fa reads.fq > tmp.sam
samtools view -b tmp.sam > out.bam
```

Create a temporary SAM file

With pipe:

```
bwa mem ref.fa reads.fq | samtools view -bS - > out.bam
```

Use "-" to specify input from STDIN instead of a file

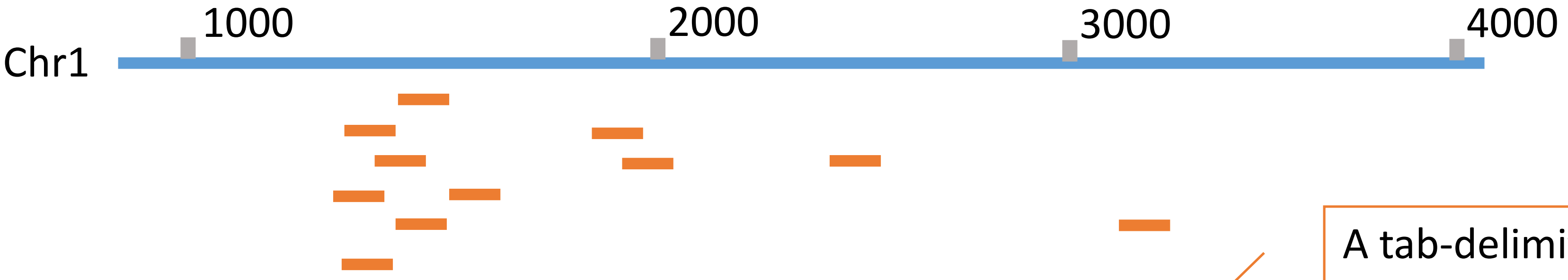
Using pipe with bed tools:

The bedtools takes in two input files, you need to specify which file from stdin

```
…… | bedtools coverage -a FirstFile -b stdin
```

Using BEDtools to process genomics data files

An example: Count the number of reads in each 1kb sliding window of the genome



A tab-delimited text file:

```
Chr1        21234023
Chr2        19282343
Chr3        15845499
```

```
bedtools makewindows -g genome.txt -w 1000 -s 1000  >  win1000.bed

bedtools coverage -abam Sample.bam -b win1000.bed -counts> coverage.bed
```

Using multi-processor machines

All BioHPC Lab machines feature <u>multiple CPU cores</u>

❑ general (cbsum1c*b*): 8 CPU cores
❑ medium-memory (cbsumm*): 24 CPU cores
❑ marge-memory (cbsulm*): 64+ CPU cores

Using multi-processor machines

Three ways to utilize multiple CPU cores on a machine:

- ☐ Using a given program's built-in parallelization, e.g.:

```
blast+ -num_threads 8 [other options]

bowtie -p 8 [other options]
```

Typically, all CPUs work together on a single task. Non-trivial, but taken care of by the programmers.

- ☐ Simultaneously executing several programs in the background, e.g.:

```
gzip file1 &
gzip file2 &
gzip file3 &
```

- ☐ If the number of independent tasks is larger than the number of CPU cores - **use a "driver" program:**

```
/programs/bin/perlscripts/perl_fork_univ.pl
```

Multiple <u>independent</u> tasks

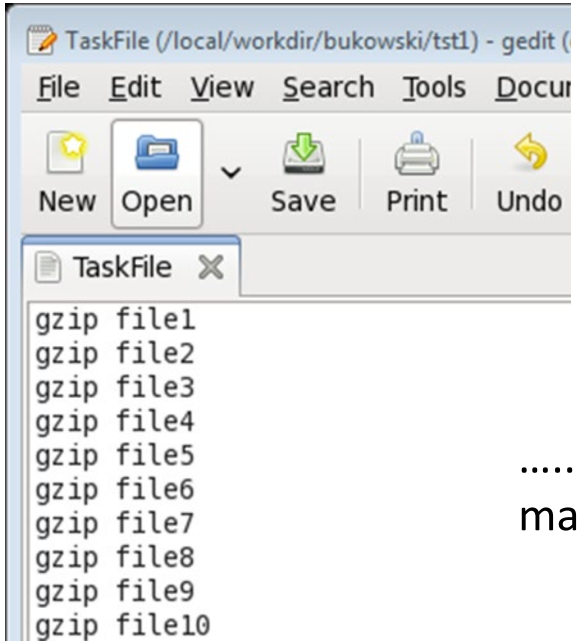
CPU: number of cores

RAM: not exceed the memory

DISK: over load the DISK IO

Using perl_fork_univ.pl

Prepare a file (called, for example, **`TaskFile`**) listing all commands to be executed. For example,



```
gzip file1
gzip file2
gzip file3
gzip file4
gzip file5
gzip file6
gzip file7
gzip file8
gzip file9
gzip file10
```

…..number of lines (i.e., tasks) may be very large

Then run the following command:

```
/programs/bin/perlscripts/perl_fork_univ.pl TaskFile NP >& log &
```

where **`NP`** is the number of processors to use (e.g., 10). The file “log” will contain some useful timing information.

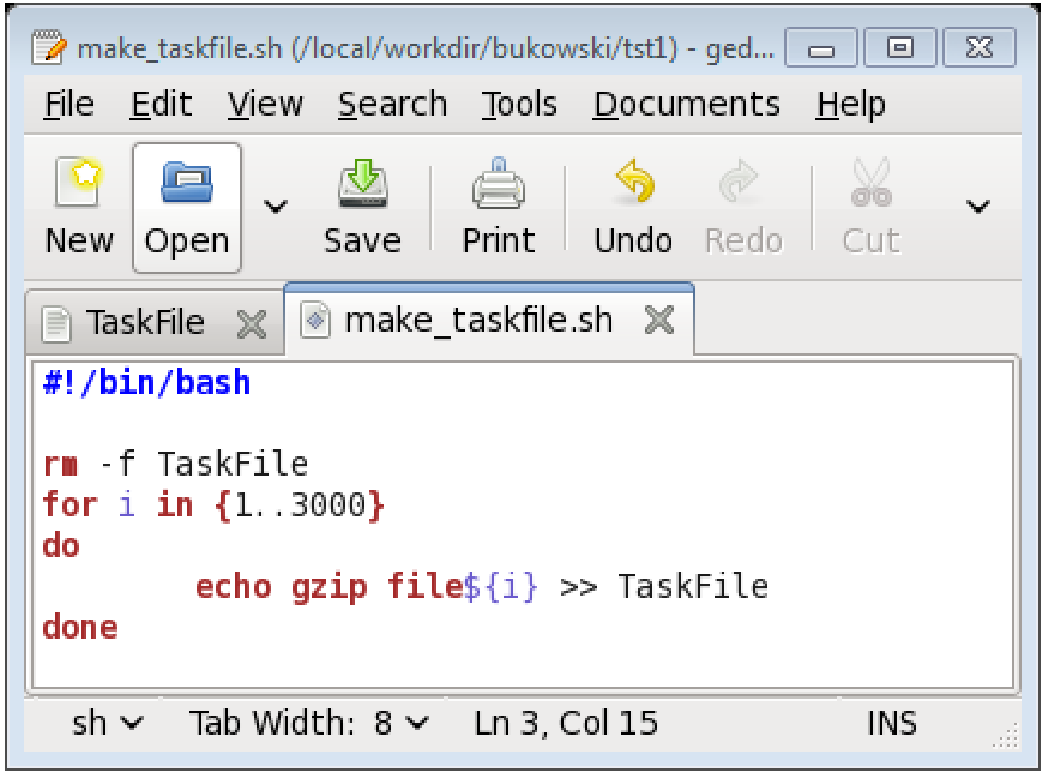
Using perl_fork_univ.pl

What does the script **`perl_fork_univ.pl`** do?

- ❑ **`perl_fork_univ.pl`** is an CBSU in-house "driver" script (written in perl)
- ❑ It will execute tasks listed in **`TaskFile`** using up to **`NP`** processors
  - The first **`NP`** tasks will be launched simultaneously
  - The **`(NP+1)`** th task will be launched right after one of the initial ones completes and a "slot" becomes available
  - The **`(NP+2)`** nd task will be launched right after another slot becomes available
  - …… etc., until all tasks are distributed
- ❑ Only up to **`NP`** tasks are running at a time (less at the end)
- ❑ All **`NP`** processors always kept busy (except near the end of task list) – **Load Balancing**

Using perl_fork_univ.pl

Ho to efficiently create a long list of tasks? Can use “**loop**” syntax built into bash:

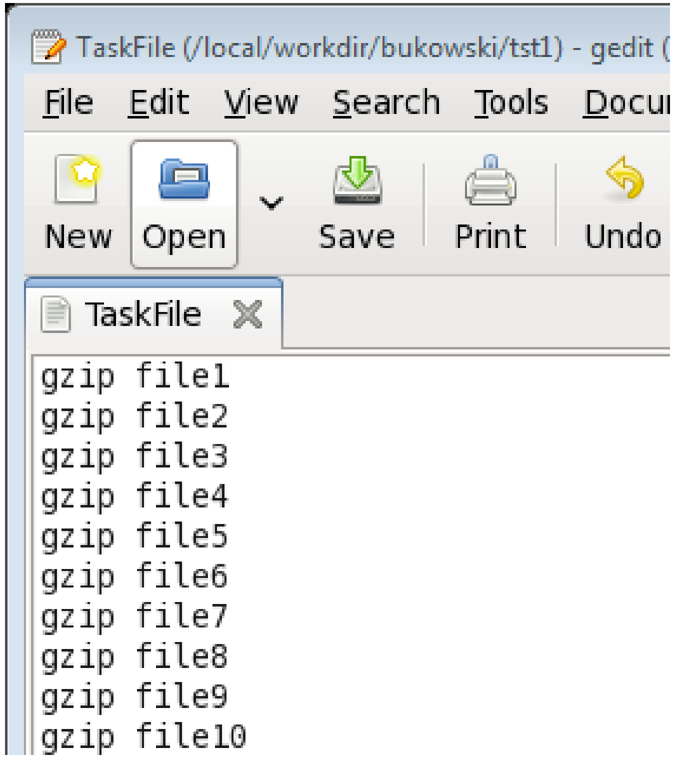


```
#!/bin/bash

rm -f TaskFile
for i in {1..3000}
do
        echo gzip file${i} >> TaskFile
done
```

sh Tab Width: 8 Ln 3, Col 15 INS

TaskFile



```
gzip file1
gzip file2
gzip file3
gzip file4
gzip file5
gzip file6
gzip file7
gzip file8
gzip file9
gzip file10
```

.....

Create an run a script like this, or just type directly from command line, ending each line with RETURN

How to choose number of CPU cores

Typically, determining the right number of CPUs to run on requires some experimentation.

Factors to consider

- total number of CPU cores on a machine: NP <= (number of CPU cores on the machine)
- combined memory required by all tasks running simultaneously should not exceed about 90% of total memory available on a machine; use **`top`** to monitor memory usage
- disk I/O bandwidth: tasks reading/write to the same disk compete for disk bandwidth. Running too many simultaneously will slow things down
- other jobs running on a machine: they also take CPU cores and memory!