Operating Systems Programming Assignment #5

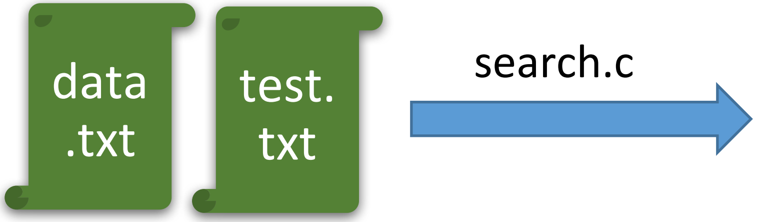
Working set optimization

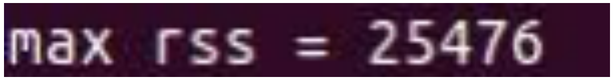
Introduction

- Working set refers to the set of pages that a process accesses
- Large working set may result in memory contention and decrease the degree of multiprogramming
- The design of data structure is closely related to the working set size
    - Reducing working set size → improving memory locality

Original Materials

- data.txt
  - Key, value pairs
- test.txt
  - queries
- search.c
  - A program execute queries

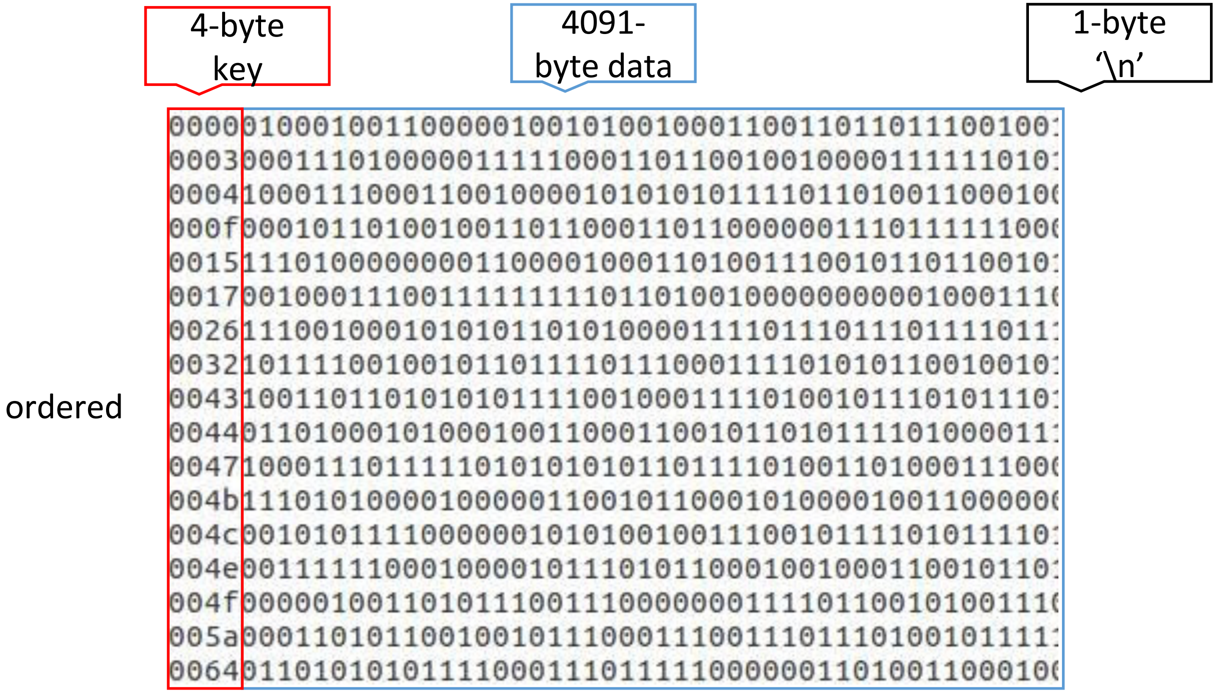




```
max rss = 25476
```

data.txt

- A large file of sorted records



4-byte key

4091-byte data

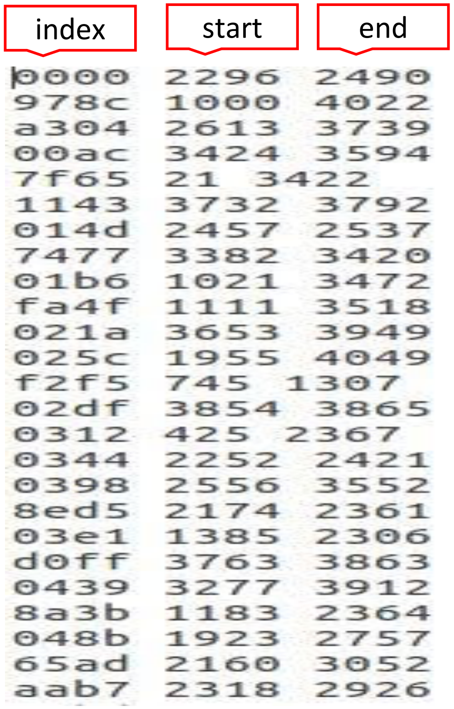
1-byte ‘\n’

ordered

```
000001000100110000001001010010001100110110111001001
000300011101000001111100011011001001000011111101011
000410001110001100100001010101011110110100110001000
000f00010110100100110110001101100000011101111110000
001511101000000001100001000110100111001011011001011
001700100011100111111111011010010000000000010001110
002611100100010101011010100001111011101110111101111
003210111100100101101111011100011110101011001001011
004310011011010101011110010001111010010111010111011
004401101000101000100110001100101101011110100001111
004710001110111110101010101101111010011010001110000
004b11101010000100000110010110001010000100110000000
004c00101011110000001010100100111001011110101111011
004e00111111000100001011101011000100100011001011011
004f00000100110101110011100000001111011001010011100
005a00011010110010010111000111001110111010010111111
006401101010101111000111011111100000011010011000100
```

test.txt format

- <key> <start> <end>\n
- <key> <start > <end>\n
- <key> <start > <end>\n
- ……

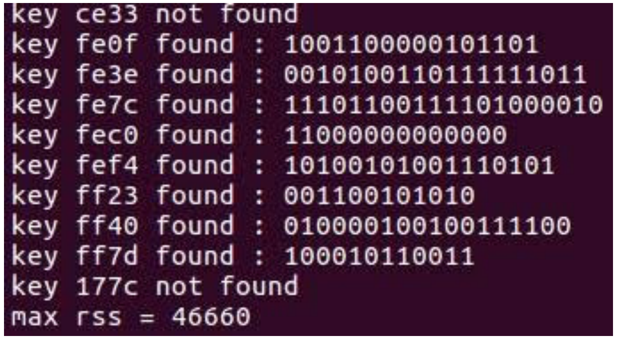


```
0000 2296 2490
978c 1000 4022
a304 2613 3739
00ac 3424 3594
7f65 21 3422
1143 3732 3792
014d 2457 2537
7477 3382 3420
01b6 1021 3472
fa4f 1111 3518
021a 3653 3949
025c 1955 4049
f2f5 745 1307
02df 3854 3865
0312 425 2367
0344 2252 2421
0398 2556 3552
8ed5 2174 2361
03e1 1385 2306
d0ff 3763 3863
0439 3277 3912
8a3b 1183 2364
048b 1923 2757
65ad 2160 3052
aab7 2318 2926
```

- E.g., f2f5 645 660
- Find key f2f5 and print characters between byte positions <start> and <end>

search.c

- Map data.txt to memory using mmap()
- Answer queries in test.txt using binary search
- Output:
  - For each query
    - Found → printf(“key %s found : %s\n”, key, buffer);
    - Not found→printf(“key %s not found\n” , key);
  - Last output
    - printf(“max rss = %lu KB\n”, ru_maxrss);

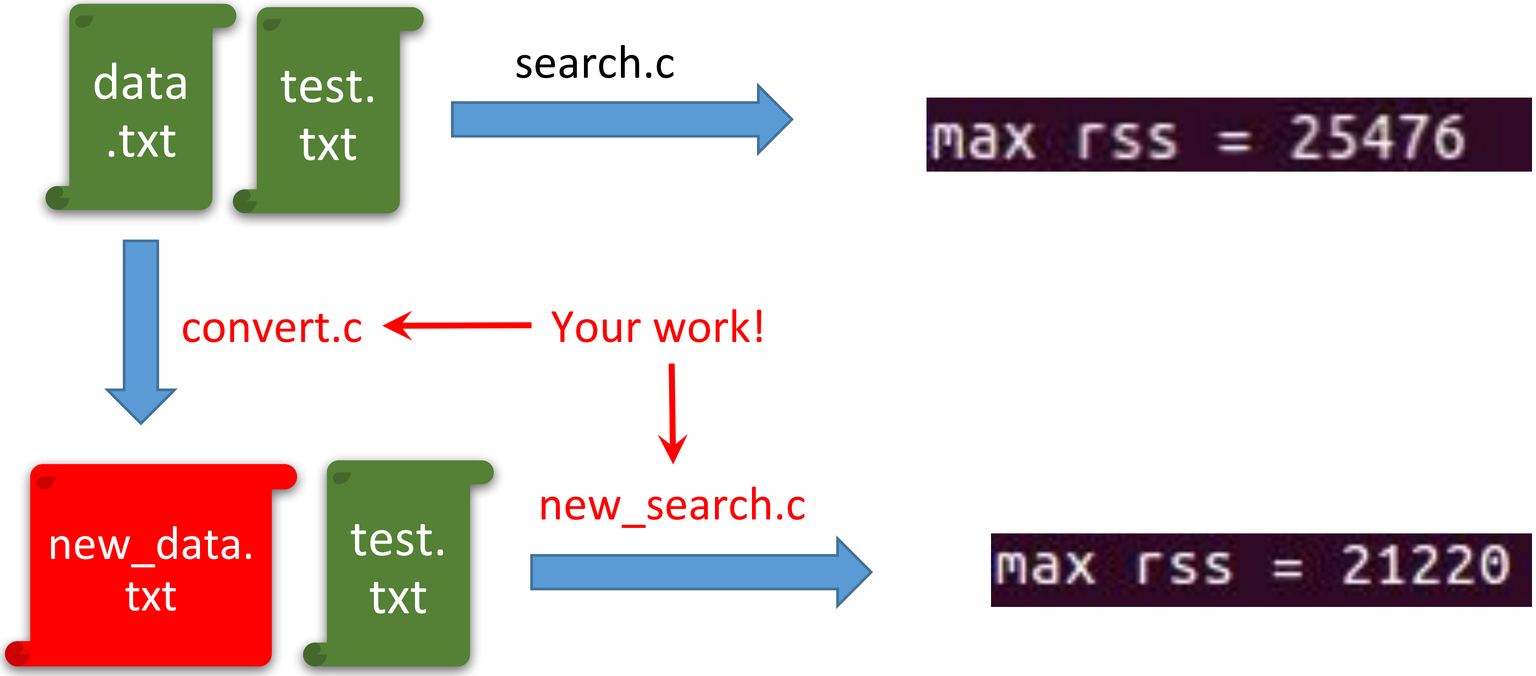


```
key ce33 not found
key fe0f found : 1001100000101101
key fe3e found : 00101001101111111011
key fe7c found : 111011001111101000010
key fec0 found : 11000000000000
key fef4 found : 10100101001110101
key ff23 found : 001100101010
key ff40 found : 010000100100111100
key ff7d found : 100010110011
key 177c not found
max rss = 46660
```

Problem Statement

- Write two programs
- convert.c
    - Convert data.txt to new_data.txt
- new_search.c
    - Search new_data.txt using the test cases in test.txt
- Your new program (new_search.c) should have smaller working set compared to the search.c!
    - At least 2000 KB reduction

Procedure



data.txt

test.txt

search.c

```
max rss = 25476
```

convert.c

Your work!

new_data.txt

test.txt

new_search.c

```
max rss = 21220
```

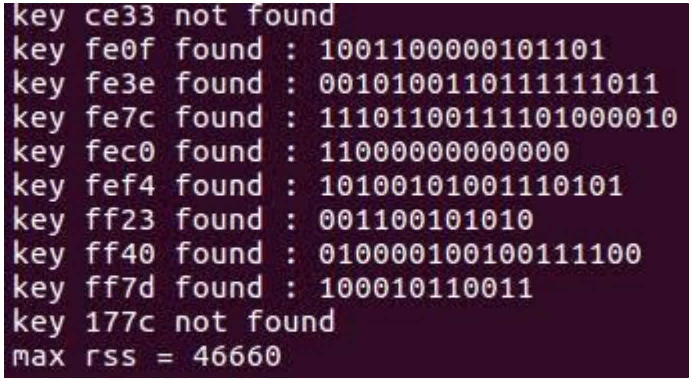
RSS (Resident Set Size):
How much physical memory is occupied by a process

convert.c

- Generate new_data.txt
  - Convert data.txt into your own file format
  - Do not modify the keys and values in data.txt!

new_search.c

- Implement your search for new_data.txt
  - Search in the memory space mapped to new_data.txt as the original *search.c* does
  - mmap(), munmap()
- Output format must be the same as search.c
  - Refer to slide No. 6
- Must produce a reduced rss



```
key ce33 not found
key fe0f found : 1001100000101101
key fe3e found : 0010100110111111011
key fe7c found : 11101100111101000010
key fec0 found : 1100000000000
key fef4 found : 1010010100111010 1
key ff23 found : 001100101010
key ff40 found : 0100001001001111 00
key ff7d found : 100010110011
key 177c not found
max rss = 46660
```

Useful APIs

- <resources.h>
  - getrusage()
    - long   ru_maxrss;   /* maximum resident set size , unit : KB*/
- <sys/mman.h>
  - mmap(), munmap()
  - Mapping a file into the logical memory space

Grading Policy

- Output Correctness
  - Must be the same format as search.c (slide No. 6)
  - TAs will test your code with another testcase.
- Search in the memory space mapped to new_data.txt
  - mmap(), munmap()
- RSS reduction
  - ru_max rss should be decreased by at least 2000KB!

Grading Policy

- Uploaded File to E3
  - student_number.zip
    - convert.c/cpp
    - new_search.c/cpp
  - Please follow the naming convention
  - Wrong file formats incur -10 pts score penalty
- Do not plagiarize

Testing OS Environment

- Ubuntu 16.04, Ubuntu 14.04 or CS linux work station
  - gcc convert.c -o convert
  - gcc new_search.c -o new_search
- Your code should compile successfully in one of the above OS environments