

Course Survey

ECE7995: Caching and Prefetching Techniques in Computer Systems

Note:

- (a) The results of the survey will be used to make any necessary adjustments of course contents.
- (b) Students are not necessarily assumed to answer 'yes' to all the questions so as to take the course.
- (c) You are voluntary to provide your name in the survey.

- 1) Have you taken any operating system (OS) or other OS-related courses before?
- 2) Have you taken any computer organization and architecture or other related courses?
- 3) Have you taken any C programming courses, or courses about data structure and algorithms design?
- 4) Approximately how many lines of C code have you written before?
- 5) Do you know that memory (or data structure as well as rights to accessing it) are separated into user space and kernel space?
- 6) Do you know for each opened file, there is an offset associated with it to indicate next read/write position in the file?
- 7) Do you know that there is a memory hierarchy in a computer system? If yes, can you list the components in the hierarchy, from the one that is closest to processor?
- 8) What do you anticipate to master after taking the course? List the items in the priority from high to low: (1) caching and prefetching algorithms (2) implementation techniques in a particular system level (3) hands-on experience with Linux kernel.
- 9) How much students' involvement in the course do you think are appropriate? More lectures given by instructor or more students' presentations/discussions?
- 10) Please write any of your thoughts, expectations and suggestions about the course.

(11) The following code is adapted from Linux kernel source code:

```
typedef long long size_t;
typedef long long loff_t;
typedef long long ssize_t

struct file_operations {
    loff_t loff_t (*llseek) (struct file *, loff_t, int);
    ssize_t (*read) (struct file *, char *, size_t, loff_t *);
    ssize_t (*write) (struct file *, const char __user *, size_t, loff_t *);
}

struct file {
    loff_t f_pos;
    unsigned int f_flags;
    struct file_operations *f_op;
    struct file *next;
}
```

(A) Search a “struct file” array to find the number of elements whose ‘f_pos’ is larger than 1024. Complete the following function, which returns the number.

```
#define SIZE 1000
struct file user_file[SIZE];

int search_files(struct file user_file[ ])
{

}

}
```

(B) Search a “struct file” linked list to find the number of elements whose ‘f_pos’ is larger than 1024. The list is linked with the “next” field of “struct file”. In addition, delete all the “struct file” nodes of the list whose ‘f_pos’ is larger than 1024, assuming that the head node has a ‘f_pos’ smaller than 1024. Please complete the following function, which returns the number and carry out the deletion operation.

```
struct file *user_file;

/* “user_file” points to the head node */
int search_files(struct file *user_file)
{

}

}
```

(C) We have a linked list described in (B). Traverse the list. For each node in the list, “struct file *f”, apply the “llseek” function with its parameters as (f, 0, 0). Complete the following function.

```
struct file *user_file;

/* “user_file” points to the head node */
void update_fpos(struct file *user_file)
{

}

}
```