

システムプログラミング演習7

学籍番号：201420694

氏名：星 遼平

1 演習7 - 1

プログラム

```
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <pthread.h>

pthread_mutex_t lock = PTHREAD_MUTEX_INITIALIZER;

char *num_str[] = { "one", "two", "three", "four", "five", "six", "seven",
    "eight", "nine", "ten", "eleven", "twelve", "thirteen", "fourteen",
    "fifteen", "sixteen", "seventeen", "eighteen", "nineteen", "twenty",
    "twenty one", "twenty two", "twenty three", "twenty four", "twenty five",
    "twenty six", "twenty seven", "twenty eight", "twenty nine", "thirty" };

int dequeue_count = 0;
int enqueue_count = 0;

struct entry {
    struct entry *next;
    void *data;
};

struct list {
    struct entry *head;
    struct entry **tail;
    pthread_cond_t notempty;
    int size;
};
```

プログラム (続き)

```
struct list *list_init(void) {
    struct list *list;

    list = malloc(sizeof *list);
    if (list == NULL)
        return (NULL);
    list->head = NULL;
    list->tail = &list->head;
    list->size = 0;
    return (list);
}

int list_enqueue(struct list *list, void *data) {
    struct entry *e;

    e = malloc(sizeof *e);
    if (e == NULL)
        return (1);
    e->next = NULL;
    e->data = data;
    pthread_mutex_lock(&lock);
    *list->tail = e;
    list->tail = &e->next;
    list->size++;
    pthread_cond_signal(&list->notempty);
    pthread_mutex_unlock(&lock);
    return (0);
}
```

プログラム (続き)

```
struct entry *list_dequeue(struct list *list) {
    struct entry *e;

    pthread_mutex_lock(&lock);
    while (list->head == NULL)
        pthread_cond_wait(&list->notempty, &lock);
    if (list->head == NULL) {
        pthread_mutex_unlock(&lock);
        return(NULL);
    }
    e = list->head;
    list->head = e->next;
    list->size--;
    if (list->head == NULL)
        list->tail = &list->head;
    pthread_mutex_unlock(&lock);
    return (e);
}
```

プログラム (続き)

```
struct entry *list_traverse
(struct list *list, int (*func)(void *, void *), void *user) {
    struct entry **prev, *n, *next;

    if (list == NULL)
        return (NULL);

    prev = &list->head;
    for (n = list->head; n != NULL; n = next) {
        next = n->next;
        switch (func(n->data, user)) {
            case 0:
                /* continues */
                prev = &n->next;
                break;
            case 1:
                /* delete the entry */
                *prev = next;
                if (next == NULL)
                    list->tail = prev;
                return (n);
            case -1:
            default:
                /* traversal stops */
                return (NULL);
        }
    }
    return (NULL);
}

int print_entry(void *e, void *u) {
    printf("%s\n", (char *)e);
    return (0);
}

int delete_entry(void *e, void *u) {
    char *c1 = e, *c2 = u;

    return (!strcmp(c1, c2));
}
```

プログラム (続き)

```
// NOTE: 30 エントリ追加するための関数
void *put_thirty_entries(void *arg) {
    int i = 0;
    struct list *list = (struct list *)arg;
    for (i = 0; i < 30; i++) {
        enqueue_count++;
        list_enqueue(list, strdup(num_str[i]));
        printf("Enqueue %d 回目 : %s\n", enqueue_count, num_str[i]);
    }
}

// NOTE: 10 エントリ取り出す関数
void *get_ten_entries(void *arg) {
    struct list *list = (struct list *)arg;
    struct entry *entry;
    int i = 0;

    for (i = 0; i < 10; i++) {
        entry = list_dequeue(list);
        dequeue_count++;
        printf("Dequeue %d 回目 : %s\n", dequeue_count, (char *)entry->data);
        free(entry->data);
        free(entry);
    }
}
```

プログラム (続き)

```
int main() {
    struct list *list;
    struct entry *entry;

    // NOTE: 30 エントリ追加するためのスレッド 2 つ
    pthread_t put_thirty_entries1, put_thirty_entries2;
    // NOTE: 10 エントリ取り出すためのスレッド 6 つ
    pthread_t get_ten_entries1, get_ten_entries2, get_ten_entries3,
              get_ten_entries4, get_ten_entries5, get_ten_entries6;

    list = list_init();

    pthread_create(&put_thirty_entries1, NULL, put_thirty_entries, (void *) (list));
    pthread_create(&get_ten_entries1, NULL, get_ten_entries, (void *) (list));
    pthread_create(&get_ten_entries2, NULL, get_ten_entries, (void *) (list));
    pthread_create(&get_ten_entries3, NULL, get_ten_entries, (void *) (list));
    pthread_create(&get_ten_entries4, NULL, get_ten_entries, (void *) (list));
    pthread_create(&get_ten_entries5, NULL, get_ten_entries, (void *) (list));
    pthread_create(&put_thirty_entries2, NULL, put_thirty_entries, (void *) (list));
    pthread_create(&get_ten_entries6, NULL, get_ten_entries, (void *) (list));

    pthread_join(put_thirty_entries1, NULL);
    pthread_join(get_ten_entries1, NULL);
    pthread_join(get_ten_entries2, NULL);
    pthread_join(get_ten_entries3, NULL);
    pthread_join(get_ten_entries4, NULL);
    pthread_join(get_ten_entries5, NULL);
    pthread_join(put_thirty_entries2, NULL);
    pthread_join(get_ten_entries6, NULL);

    /* entry list */
    list_traverse(list, print_entry, NULL);

    return (0);
}
```

実行結果

```
$ ./app
Enqueue 1 回目 : one
Enqueue 2 回目 : two
Enqueue 3 回目 : three
Enqueue 4 回目 : four
Enqueue 5 回目 : five
Enqueue 6 回目 : six
Enqueue 7 回目 : seven
Enqueue 8 回目 : eight
Enqueue 9 回目 : nine
Enqueue 10 回目 : ten
Enqueue 11 回目 : eleven
Enqueue 12 回目 : twelve
Enqueue 13 回目 : thirteen
Enqueue 14 回目 : fourteen
Enqueue 15 回目 : fifteen
Enqueue 16 回目 : sixteen
Enqueue 17 回目 : seventeen
Enqueue 18 回目 : eighteen
Enqueue 19 回目 : nineteen
Enqueue 20 回目 : twenty
Enqueue 21 回目 : twenty one
Enqueue 22 回目 : twenty two
Enqueue 23 回目 : twenty three
Enqueue 24 回目 : twenty four
Enqueue 25 回目 : twenty five
Enqueue 26 回目 : twenty six
Enqueue 27 回目 : twenty seven
Enqueue 28 回目 : twenty eight
Enqueue 29 回目 : twenty nine
Enqueue 30 回目 : thirty
Dequeue 1 回目 : one
Dequeue 4 回目 : four
Dequeue 5 回目 : five
Dequeue 6 回目 : six
Dequeue 7 回目 : seven
Dequeue 8 回目 : eight
Dequeue 9 回目 : nine
Dequeue 10 回目 : ten
Dequeue 11 回目 : eleven
Dequeue 12 回目 : twelve
```

実行結果 (続き)

Dequeue 3 回目 : three
Dequeue 13 回目 : thirteen
Dequeue 14 回目 : fourteen
Dequeue 15 回目 : fifteen
Dequeue 16 回目 : sixteen
Dequeue 17 回目 : seventeen
Dequeue 18 回目 : eighteen
Dequeue 19 回目 : nineteen
Dequeue 20 回目 : twenty
Dequeue 21 回目 : twenty one
Dequeue 2 回目 : two
Dequeue 22 回目 : twenty two
Dequeue 23 回目 : twenty three
Dequeue 24 回目 : twenty four
Dequeue 25 回目 : twenty five
Dequeue 26 回目 : twenty six
Dequeue 27 回目 : twenty seven
Dequeue 28 回目 : twenty eight
Dequeue 29 回目 : twenty nine
Dequeue 30 回目 : thirty
Dequeue 31 回目 : one
Enqueue 31 回目 : one
Dequeue 32 回目 : two
Enqueue 32 回目 : two
Enqueue 33 回目 : three
Dequeue 33 回目 : three
Dequeue 34 回目 : four
Enqueue 34 回目 : four
Dequeue 35 回目 : five
Enqueue 35 回目 : five
Dequeue 36 回目 : six
Enqueue 36 回目 : six
Dequeue 37 回目 : seven
Enqueue 37 回目 : seven
Dequeue 38 回目 : eight
Enqueue 38 回目 : eight
Dequeue 39 回目 : nine
Enqueue 39 回目 : nine
Dequeue 40 回目 : ten
Enqueue 40 回目 : ten
Dequeue 41 回目 : eleven
Enqueue 41 回目 : eleven

実行結果 (続き)

Dequeue 42 回目 : twelve
Enqueue 42 回目 : twelve
Dequeue 43 回目 : thirteen
Enqueue 43 回目 : thirteen
Dequeue 44 回目 : fourteen
Enqueue 44 回目 : fourteen
Dequeue 45 回目 : fifteen
Enqueue 45 回目 : fifteen
Dequeue 46 回目 : sixteen
Enqueue 46 回目 : sixteen
Dequeue 47 回目 : seventeen
Enqueue 47 回目 : seventeen
Dequeue 48 回目 : eighteen
Enqueue 48 回目 : eighteen
Dequeue 49 回目 : nineteen
Enqueue 49 回目 : nineteen
Dequeue 50 回目 : twenty
Enqueue 50 回目 : twenty
Dequeue 51 回目 : twenty one
Enqueue 51 回目 : twenty one
Dequeue 52 回目 : twenty two
Enqueue 52 回目 : twenty two
Dequeue 53 回目 : twenty three
Enqueue 53 回目 : twenty three
Dequeue 54 回目 : twenty four
Enqueue 54 回目 : twenty four
Dequeue 55 回目 : twenty five
Enqueue 55 回目 : twenty five
Dequeue 56 回目 : twenty six
Enqueue 56 回目 : twenty six
Dequeue 57 回目 : twenty seven
Enqueue 57 回目 : twenty seven
Dequeue 58 回目 : twenty eight
Enqueue 58 回目 : twenty eight
Dequeue 59 回目 : twenty nine
Enqueue 59 回目 : twenty nine
Dequeue 60 回目 : thirty
Enqueue 60 回目 : thirty

2 演習 7-2

プログラム

```
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <pthread.h>

char *num_str[] = { "one", "two", "three", "four", "five", "six", "seven",
    "eight", "nine", "ten", "eleven", "twelve", "thirteen", "fourteen",
    "fifteen", "sixteen", "seventeen", "eighteen", "nineteen", "twenty",
    "twenty one", "twenty two", "twenty three", "twenty four", "twenty five",
    "twenty six", "twenty seven", "twenty eight", "twenty nine", "thirty" };

int dequeue_count = 0;
int enqueue_count = 0;

struct entry {
    struct entry *next;
    void *data;
};

struct list {
    struct entry *head;
    struct entry **tail;
    pthread_mutex_t lock;
    int size;
};

struct list *list_init(void) {
    struct list *list;

    list = malloc(sizeof *list);
    if (list == NULL)
        return (NULL);
    list->head = NULL;
    list->tail = &list->head;
    list->lock = PTHREAD_MUTEX_INITIALIZER;
    list->size = 0;
    return (list);
}
```

プログラム (続き)

```
int list_enqueue(struct list *list, void *data) {
    struct entry *e;

    e = malloc(sizeof *e);
    if (e == NULL)
        return (1);
    e->next = NULL;
    e->data = data;
    pthread_mutex_lock(&list->lock);
    *list->tail = e;
    list->tail = &e->next;
    list->size++;
    pthread_mutex_unlock(&list->lock);
    return (0);
}

struct entry *list_dequeue(struct list *list) {
    struct entry *e;

    pthread_mutex_lock(&list->lock);
    if (list->head == NULL) {
        pthread_mutex_unlock(&list->lock);
        return(NULL);
    }
    e = list->head;
    list->head = e->next;
    list->size--;
    if (list->head == NULL)
        list->tail = &list->head;
    pthread_mutex_unlock(&list->lock);
    return (e);
}
```

プログラム (続き)

```
struct entry *list_traverse
(struct list *list, int (*func)(void *, void *), void *user) {
    struct entry **prev, *n, *next;

    if (list == NULL)
        return (NULL);

    prev = &list->head;
    for (n = list->head; n != NULL; n = next) {
        next = n->next;
        switch (func(n->data, user)) {
            case 0:
                /* continues */
                prev = &n->next;
                break;
            case 1:
                /* delete the entry */
                *prev = next;
                if (next == NULL)
                    list->tail = prev;
                return (n);
            case -1:
            default:
                /* traversal stops */
                return (NULL);
        }
    }
    return (NULL);
}
```

プログラム (続き)

```
int print_entry(void *e, void *u) {
    printf("%s\n", (char *)e);
    return (0);
}

int delete_entry(void *e, void *u) {
    char *c1 = e, *c2 = u;

    return (!strcmp(c1, c2));
}

// NOTE: 30 エントリ追加するための関数
void *put_thirty_entries(void *arg) {
    int i = 0;
    struct list *list = (struct list *)arg;
    for (i = 0; i < 30; i++) {
        enqueue_count++;
        list_enqueue(list, strdup(num_str[i]));
        printf("Enqueue %d 回目 : %s\n", enqueue_count, num_str[i]);
    }
}

// NOTE: 10 エントリ取り出す関数
void *get_ten_entries(void *arg) {
    struct list *list = (struct list *)arg;
    struct entry *entry;
    int i = 0;

    for (i = 0; i < 10; i++) {
        entry = list_dequeue(list);
        dequeue_count++;
        printf("Dequeue %d 回目 : %s\n", dequeue_count, (char *)entry->data);
        free(entry->data);
        free(entry);
    }
}
```

プログラム (続き)

```
int main() {
    struct list *list;
    struct entry *entry;

    // NOTE: 30 エントリ追加するためのスレッド 2 つ
    pthread_t put_thirty_entries1, put_thirty_entries2;
    // NOTE: 10 エントリ取り出すためのスレッド 6 つ
    pthread_t get_ten_entries1, get_ten_entries2, get_ten_entries3,
              get_ten_entries4, get_ten_entries5, get_ten_entries6;

    list = list_init();

    pthread_create(&put_thirty_entries1, NULL, put_thirty_entries, (void *) (list));
    pthread_create(&get_ten_entries1, NULL, get_ten_entries, (void *) (list));
    pthread_create(&get_ten_entries2, NULL, get_ten_entries, (void *) (list));
    pthread_create(&get_ten_entries3, NULL, get_ten_entries, (void *) (list));
    pthread_create(&get_ten_entries4, NULL, get_ten_entries, (void *) (list));
    pthread_create(&get_ten_entries5, NULL, get_ten_entries, (void *) (list));
    pthread_create(&put_thirty_entries2, NULL, put_thirty_entries, (void *) (list));
    pthread_create(&get_ten_entries6, NULL, get_ten_entries, (void *) (list));

    pthread_join(put_thirty_entries1, NULL);
    pthread_join(get_ten_entries1, NULL);
    pthread_join(get_ten_entries2, NULL);
    pthread_join(get_ten_entries3, NULL);
    pthread_join(get_ten_entries4, NULL);
    pthread_join(get_ten_entries5, NULL);
    pthread_join(put_thirty_entries2, NULL);
    pthread_join(get_ten_entries6, NULL);

    /* entry list */
    list_traverse(list, print_entry, NULL);

    return (0);
}
```

実行結果

```
$ ./app
Enqueue 1 回目 : one
Enqueue 2 回目 : two
Enqueue 3 回目 : three
Enqueue 4 回目 : four
Enqueue 5 回目 : five
Enqueue 6 回目 : six
Enqueue 7 回目 : seven
Enqueue 8 回目 : eight
Enqueue 9 回目 : nine
Enqueue 10 回目 : ten
Enqueue 11 回目 : eleven
Enqueue 12 回目 : twelve
Enqueue 13 回目 : thirteen
Enqueue 14 回目 : fourteen
Enqueue 15 回目 : fifteen
Enqueue 16 回目 : sixteen
Enqueue 17 回目 : seventeen
Enqueue 18 回目 : eighteen
Enqueue 19 回目 : nineteen
Enqueue 20 回目 : twenty
Enqueue 21 回目 : twenty one
Enqueue 22 回目 : twenty two
Enqueue 23 回目 : twenty three
Enqueue 24 回目 : twenty four
Enqueue 25 回目 : twenty five
Enqueue 26 回目 : twenty six
Enqueue 27 回目 : twenty seven
Enqueue 28 回目 : twenty eight
Enqueue 29 回目 : twenty nine
Enqueue 30 回目 : thirty
Dequeue 1 回目 : one
Dequeue 4 回目 : four
Dequeue 5 回目 : five
Dequeue 6 回目 : six
Dequeue 7 回目 : seven
Dequeue 8 回目 : eight
Dequeue 9 回目 : nine
Dequeue 10 回目 : ten
Dequeue 11 回目 : eleven
Dequeue 12 回目 : twelve
```

実行結果 (続き)

Dequeue 3 回目 : three
Dequeue 13 回目 : thirteen
Dequeue 14 回目 : fourteen
Dequeue 15 回目 : fifteen
Dequeue 16 回目 : sixteen
Dequeue 17 回目 : seventeen
Dequeue 18 回目 : eighteen
Dequeue 19 回目 : nineteen
Dequeue 20 回目 : twenty
Dequeue 21 回目 : twenty one
Dequeue 2 回目 : two
Dequeue 22 回目 : twenty two
Dequeue 23 回目 : twenty three
Dequeue 24 回目 : twenty four
Dequeue 25 回目 : twenty five
Dequeue 26 回目 : twenty six
Dequeue 27 回目 : twenty seven
Dequeue 28 回目 : twenty eight
Dequeue 29 回目 : twenty nine
Dequeue 30 回目 : thirty
Dequeue 31 回目 : one
Enqueue 31 回目 : one
Dequeue 32 回目 : two
Enqueue 32 回目 : two
Enqueue 33 回目 : three
Dequeue 33 回目 : three
Dequeue 34 回目 : four
Enqueue 34 回目 : four
Dequeue 35 回目 : five
Enqueue 35 回目 : five
Dequeue 36 回目 : six
Enqueue 36 回目 : six
Dequeue 37 回目 : seven
Enqueue 37 回目 : seven
Dequeue 38 回目 : eight
Enqueue 38 回目 : eight
Dequeue 39 回目 : nine
Enqueue 39 回目 : nine
Dequeue 40 回目 : ten
Enqueue 40 回目 : ten
Dequeue 41 回目 : eleven
Enqueue 41 回目 : eleven

実行結果 (続き)

Dequeue 42 回目 : twelve
Enqueue 42 回目 : twelve
Dequeue 43 回目 : thirteen
Enqueue 43 回目 : thirteen
Dequeue 44 回目 : fourteen
Enqueue 44 回目 : fourteen
Dequeue 45 回目 : fifteen
Enqueue 45 回目 : fifteen
Dequeue 46 回目 : sixteen
Enqueue 46 回目 : sixteen
Dequeue 47 回目 : seventeen
Enqueue 47 回目 : seventeen
Dequeue 48 回目 : eighteen
Enqueue 48 回目 : eighteen
Dequeue 49 回目 : nineteen
Enqueue 49 回目 : nineteen
Dequeue 50 回目 : twenty
Enqueue 50 回目 : twenty
Dequeue 51 回目 : twenty one
Enqueue 51 回目 : twenty one
Dequeue 52 回目 : twenty two
Enqueue 52 回目 : twenty two
Dequeue 53 回目 : twenty three
Enqueue 53 回目 : twenty three
Dequeue 54 回目 : twenty four
Enqueue 54 回目 : twenty four
Dequeue 55 回目 : twenty five
Enqueue 55 回目 : twenty five
Dequeue 56 回目 : twenty six
Enqueue 56 回目 : twenty six
Dequeue 57 回目 : twenty seven
Enqueue 57 回目 : twenty seven
Dequeue 58 回目 : twenty eight
Enqueue 58 回目 : twenty eight
Dequeue 59 回目 : twenty nine
Enqueue 59 回目 : twenty nine
Dequeue 60 回目 : thirty
Enqueue 60 回目 : thirty

3 考察

スレッドが扱うデータのロック方法は様々あるが、それぞれに特徴があり使いどころを間違えないようにしないと、デッドロックの危険性やパフォーマンス低下の恐れがあるので、各データロックの特徴を把握した上で開発することが重要であるとわかった。

4 授業の感想

今回の課題を解く上で授業で説明していた内容がかなり重要になっており、授業のスライドはあくまで補足説明程度なので、次回が最後ですがメモを取るようにしようと思い直しました。

授業の最初に前回の課題内容の解答・解説がありますが、スライドに含まれていないのでメモを取る時間が少しほしいなと思いました。