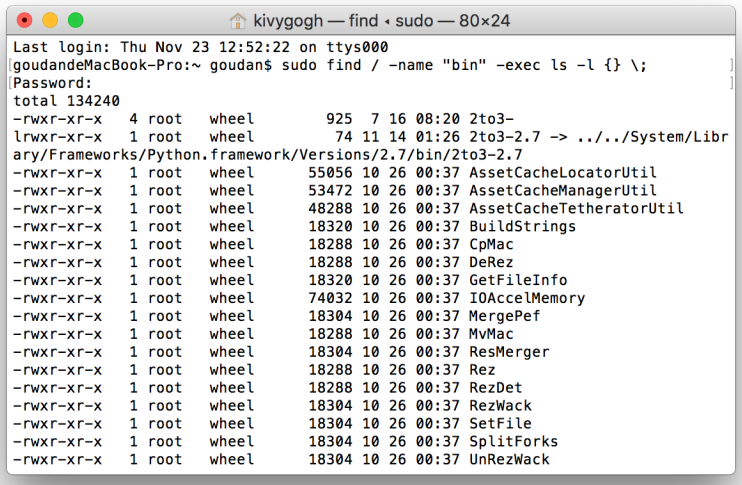
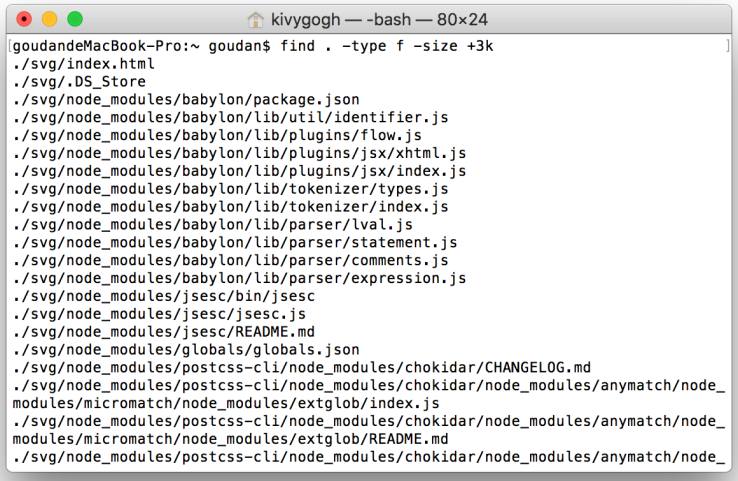
ex1:

1.sudo find ./ -name "bin" -exec ls -l {} \;



2.find ./ -type f -size +3k



3.find. / -exec o 000 chmod 755 {} \;

4.find ./ -name "*.c" root | xargs wc -l | sort -n >> sorted_statistics.txt

5.find ./ root | wc -l

ex2:

1.Use pthread_create to save threads' id, bind their thread function and send args to threads.

pthread_create(&tid[i], NULL, thread_handler, (void *)&slp);

2.Use pthread_join to waits threads' termination, and receive their return value.

pthread_join(tid[i],(void**)&exc);

3.Use getpid to get pid, use pthread_self to get thread identifier.

printf("process id: %d\tthread id: %lu\n", getpid(),(unsigned long)pthread_self());

ex702.c

ex702.c 〉 No Selection

```c
#define THREAD_NUM 1

void * thread_handler(void *arg);

int main(int argc,char *argv[]){

    if(argc<2){
        perror("The number of args is wrong!\n");
        exit(0);
    }

    int i,slp;
    int *exc;

    slp = atoi(argv[1]);

    pthread_t tid[THREAD_NUM];

    for(i = 0; i < THREAD_NUM; i++){

        pthread_create(&tid[i], NULL, thread_handler, (void *)&slp);

    }

    for(i = 0; i < THREAD_NUM; i++){

        pthread_join(tid[i],(void**)&exc);
        printf("thread id:%lu,exit code:%d\n", (unsigned long)tid[i], *exc);

    }
    return 0;
}

void *thread_handler(void *arg){
    int s = *((int*)arg);
    static int excode;
    printf("process id: %d\tthread id: %lu\n", getpid(),(unsigned long)pthread_self());
    sleep(s);
    scanf("%d",&excode);
    return (void *)(&excode);
}
```

ex3:

1.Use a binary tree to store every thread's deep and thread id. Malloc sharing memory to binary tree in main thread.

Tree root;

creat_tree(root,n);

root->tid = pthread_self();

2.Creat threads and shore datas.

root->left->deep=n-1;

pthread_create(&root->left->tid, NULL, thread_handler, root->left);

root->right->deep=n-1;

pthread_create(&root->right->tid, NULL, thread_handler, root->right);

3.Generate child threads and shore datas in the binary tree in headler function.

```
void *thread_handler(void *args){
    Tree t = (Tree)args;
    if(t->deep == 0)
    {   t->left = NULL;
        t->right = NULL;
        exit(0);
    }else{
        pthread_create(&t->left->tid, NULL, thread_handler, t->left);
        pthread_create(&t->right->tid, NULL, thread_handler, t->right);
    }
    return NULL;
}
```