Chapter 4
Threads

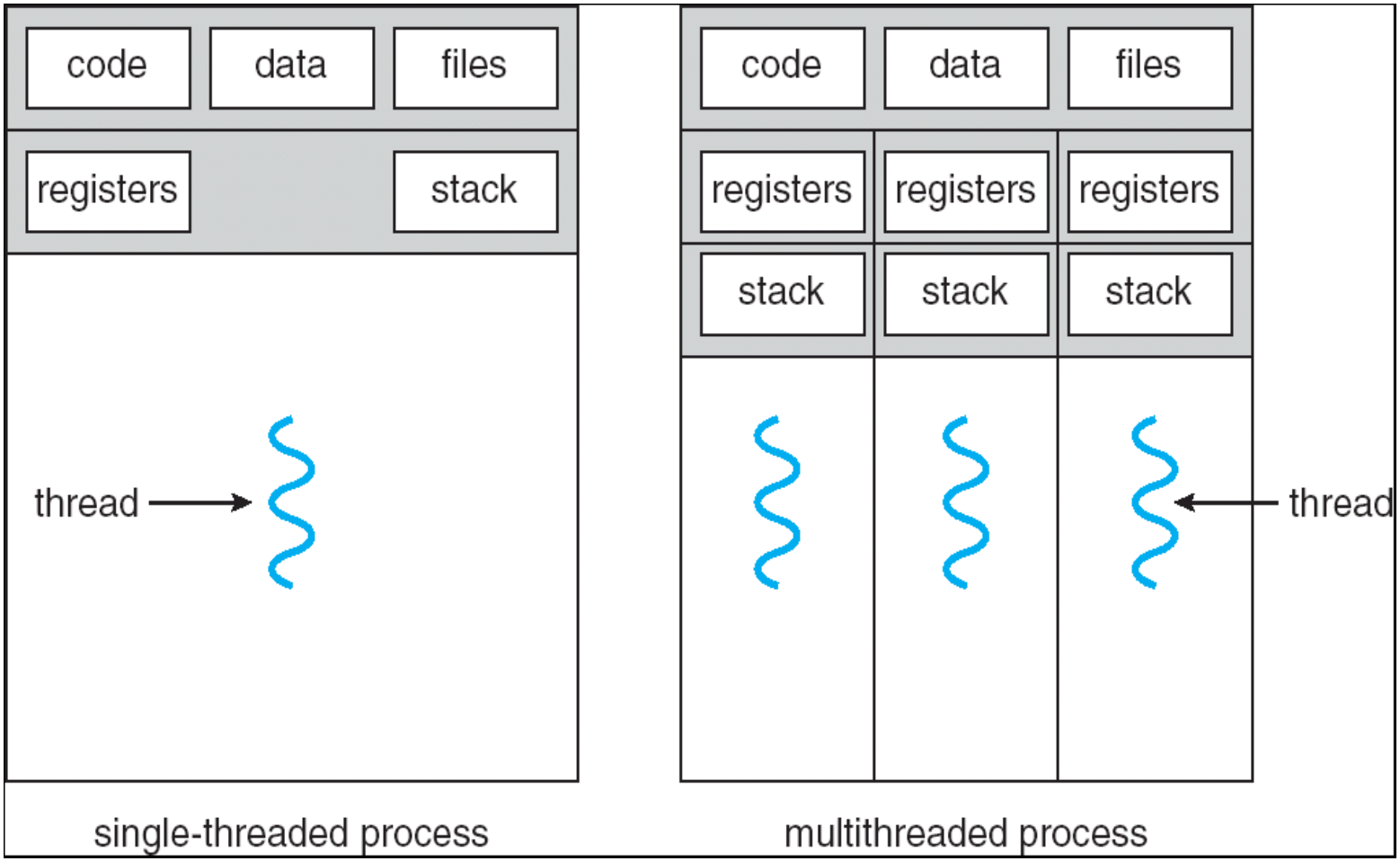
Images from Silberschatz

pipeBug.c

```
pipe(thePipe);
childPid = fork();
if (childPid == 0) {

    /* I AM A CHILD */

    while (read(thePipe[READ], data, MAXSIZE) > 0) {
        printf("CHILD> %s\n", data);
    }
    close(thePipe[READ]);
} else {

    /* parent */

    readFromCommandLine(data, MAXSIZE);
    while (strncmp(data, "STOP", 4) != 0)
    {
        write(thePipe[WRITE], data, strlen(data)+1);
        readFromCommandLine(data, MAXSIZE);
    }
    close(thePipe[WRITE]);
    waitpid(childPid, &status, 0);
}
```

Threads

- Multiple lines of control *inside one process*
  - Faster to create a thread than spawn a process
- Each has its own registers & stack
- Shared code & data



Typical Usages

- Word Processor

- Web Server

Benefits

- Why multithread?

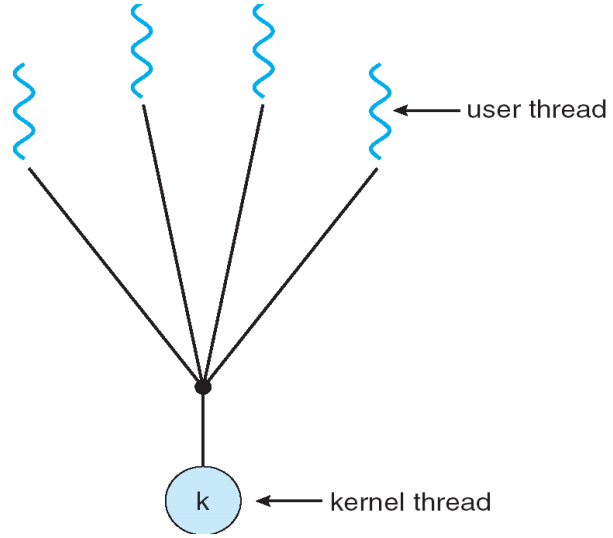
User vs Kernel Threads

- User:

- Kernel:

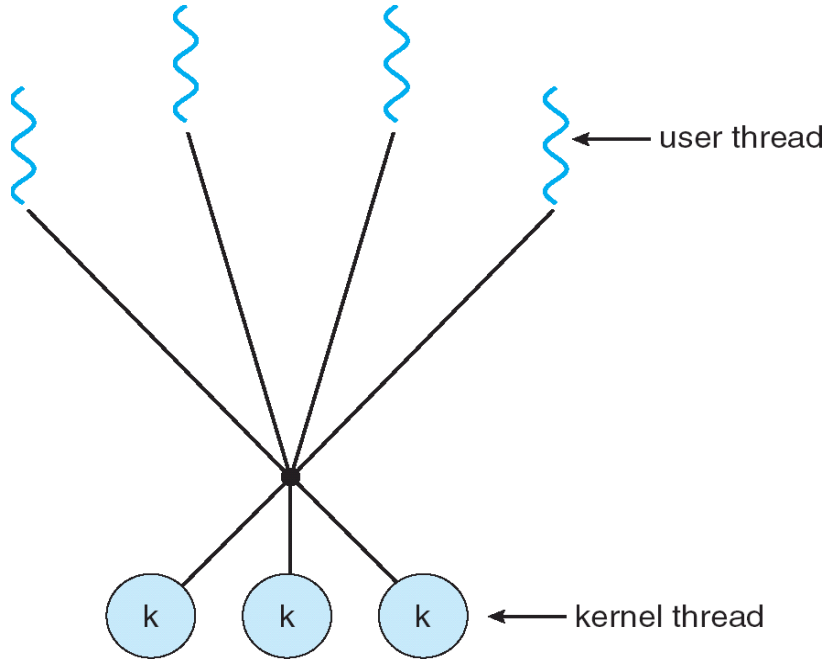
Models

- Many-to-One



- One-to-One

- Many-to-Many



blocking system call?

Thread Libraries

- User vs Kernel
- POSIX Pthreads
- Win32
- Java

Pthreads

- Linux, cygwin, Solaris, etc.
  - libpthread.so
  - gcc -g -o appName appname.c -lpthread

```c
/* This code works on Zeus!
 * link with -lpthread
 * gcc -o app -g app.o -lpthread
 */
#include <pthread.h>
#include <stdio.h>

int sum; /* this data is shared by the thread(s) */

void *runner(void *param); /* the thread */

int main(int argc, char *argv[])
{
  pthread_t tid; /* the thread identifier */
  pthread_attr_t attr; /* set of attributes for the thread */

  if (argc != 2)
  {
    fprintf(stderr,"usage: a.out <integer value>\n");
    /*exit(1);*/
    return -1;
  }

/* page 133 of Silberschatz */
```

```
  if (atoi(argv[1]) < 0)
  {
    fprintf(stderr,"Argument %d must be nonneg\n",atoi(argv[1]));
    /*exit(1);*/
    return -1;
  }

  /* get the default attributes */
  pthread_attr_init(&attr);

  /* create the thread */
  pthread_create(&tid,&attr,runner,argv[1]);

  /* now wait for the thread to exit */
  pthread_join(tid,NULL);

  printf("sum = %d\n",sum);
}
/* page 133 of Silberschatz */
```

```c
/**
 * The thread will begin control in this function
 */
void *runner(void *param)
{
  int i, upper = atoi(param);
  sum = 0;

  if (upper > 0)
  {
    for (i = 1; i <= upper; i++)
    {
      sum += i;
    }
  }

  pthread_exit(0);
}
/* page 133 of Silberschatz */
```

Mutex!

```
/* This code works on Zeus!
 * link with -lpthread
 * gcc -o app -g app.o -lpthread
 */
#include <pthread.h>
#include <stdio.h>
#define MAX 10

int sum; /* this data is shared by the thread(s) */
pthread_mutex_t gMutex;

void *runner(void *param); /* the thread */

int main(int argc, char *argv[])
{
  pthread_t tid1, tid2; /* the thread identifier */
  pthread_attr_t attr; /* set of attributes for the thread */



/* page 133 of Silberschatz */
```

```
  /* init the mutex */
  pthread_mutex_init(&gMutex, NULL);

  /* get the default attributes */
  pthread_attr_init(&attr);

  /* create the threads */
  pthread_create(&tid1,&attr,runner,&threadParamOne);
  pthread_create(&tid2,&attr,runner,&threadParamTwo);

  /* now wait for the threads to exit */
  pthread_join(tid1,NULL);
  pthread_join(tid2,NULL);

  pthread_mutex_destroy(&gMutex);
  pthread_attr_destory(&attr);

  printf("sum = %d\n",sum);
}
/* page 133 of Silberschatz */
```

```c
/**
 * The thread will begin control in this function
 */
void *runner(void *param)
{
  int i;
  sum = 0;

  for (i = 1; i <= MAX; i++)
  {
    pthread_mutex_lock(&gMutex);
    sum += i;
    pthread_mutex_unlock(&gMutex);
  }

  pthread_exit(0);
}
/* page 133 of Silberschatz */
```