# 操作系统

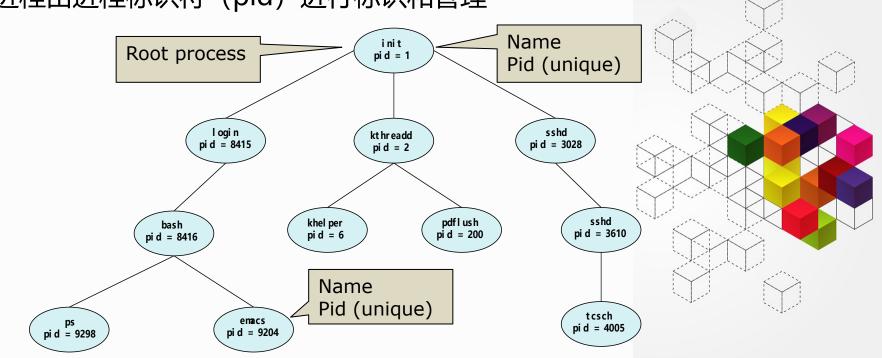
Operating system

孔维强 大连理工大学



• 父进程可生成子进程,子进程可进一步生成子进程,形成树结构

• 进程由进程标识符 (pid) 进行标识和管理

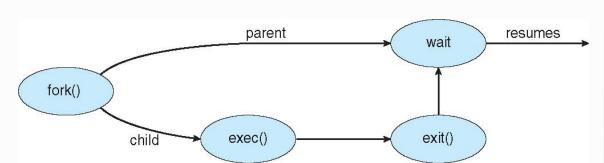


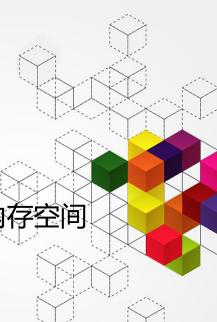
#### 地址 (内存) 空间选项

- 子进程复制父进程的地址空间(相同程序和数据)
- 子进程加载新的程序并运行

#### UNIX系统举例

- · fork()系统调用生成子进程
- exec()系统调用在fork()后执行,用新程序替代进程的内存空间





```
int value = 5;
                                                        int value = 5;
int main()
                                                        int main()
  pid t pid;
                                                          pid_t pid;
   /* fork another process */
                                                            /* fork another process */
    pid = fork();
                                                            pid = fork();
    if (pid < 0) { /* error occurred */
                                                            if (pid < 0) { /* error occurred */
           fprintf(stderr, "Fork Failed");
                                                                    fprintf(stderr, "Fork Failed");
           exit(-1); }
                                                                    exit(-1); }
    else if (pid == 0) { /* child process */
                                                            else if (pid == 0) { /* child process */
           value += 15; }
                                                                    value += 15; }
    else { /* parent process */
                                                            else { /* parent process */
             /* parent will wait for the child to
                                                                      /* parent will wait for the child to
    complete */
                                                            complete */
           wait (NULL);
                                                                    wait (NULL);
           printf ("PARENT: value = %d", value); //
                                                                    printf ("PARENT: value = %d", value); //
    LINE A
                                                            LINE A
           exit(0); } }
                                                                    exit(0); } }
```

### 线程的一些思考

#### fork()和exec()系统调用的语义

fork()生成子进程, exec()加载新的程序

如果线程调用fork()的话,fork()复制整个进程还是仅复制调用的线程?

- 一部分UNIX系统支持2种版本
- 一选择哪个版本取决于fork()后是否有exec()。 (why?)
- —exec()通常与之前一致,替换执行进程的程序,包括所有线程

```
#include <pthread.h>
                                                                 #include <pthread.h>
#include <stdio.h>
                                                                 #include <stdio.h>
int value = 0:
                                                                 int value = 0:
void *runner(void *param); /* the thread */
                                                                 void *runner(void *param); /* the thread */
int main(int argc, char *argv[])
                                                                 int main(int argc, char *argv[])
int pid:
                                                                 int pid:
pthread_t tid;
                                                                 pthread_t tid;
pthread_attr_t attr;
                                                                 pthread_attr_t attr;
  pid = fork();
                                                                    pid = fork();
  if (pid == 0) {/* child process */
                                                                    if (pid == 0) {/* child process */
                                                                      pthread_attr_init(&attr);
     pthread_attr_init(&attr);
     pthread_create(&tid,&attr,runner,NULL);
                                                                      pthread_create(&tid,&attr,runner,NULL);
    pthread_join(tid,NULL);
                                                                      pthread_join(tid,NULL);
     printf("CHILD: value = %d", value); /* LINE C */
                                                                      printf("CHILD: value = %d", value); /* LINE C */
  else if (pid > 0) {/* parent process */
                                                                    else if (pid > 0) {/* parent process */
    wait (NULL) :
                                                                      wait (NULL) :
     printf("PARENT: value = %d", value); /* LINE P */
                                                                      printf("PARENT: value = %d", value); /* LINE P */
void *runner(void *param) {
                                                                 void *runner(void *param) {
  value = 5;
                                                                   value = 5;
  pthread_exit(0);
                                                                   pthread_exit(0);
```