Programovanie v oeračných systémoch

06 - Processes, threads

Jozef Šiška

Department of Applied Informatics
Comenius University in Bratislava

2016/2017

Process

- instance of a program
- processor state, context (registers,. . .)
- virtual memory
- resources (file desc. , security info (user, group, capabilities)

- `fork()` (+ `exec(...)`)
- `exit()`
- `waitpid()`

Thread

- lightweight
- processor state, context (registers,. . .)
- shared virtual memory, resources

- `pthread_create()` (`clone(...)`)
- `pthread_exit()`
- `pthread_join()`

Example

```
void *task_hello(void *data)
{
    printf("Hello world, data: %p\n", data);
    pthread_exit(NULL);
}

int main (int argc, char *argv[])
{
    pthread_t threads[NUM_THREADS];
    int ret;
    int i;
    for (i = 0; i < NUM_THREADS; ++i) {
        printf("main: pthread_create %d\n", i);
        if ((ret = pthread_create(&threads[i], NULL, task_hello, NULL))) {
            printf("pthread_create: error: %d\n", ret);
            exit(EXIT_FAILURE);
        }
    }

    for (i = 0; i < NUM_THREADS; ++i) {
        void *retval = NULL;
        if ((ret = pthread_join(threads[i], &retval) != 0)) {
            printf("pthread_join: error %d\n", ret);
        } else {
            printf("thread %d finished with return value %p\n", i, retval);
        }
    }
    pthread_exit(NULL);
}
```

Comparison

	process	thread
create	`fork()`	`pthread_create()`
exit	`exit(int status)`	`pthread_exit(void *retval)`
wait	`waitpid()`	`pthread_join()`
identification	unique PID	shared PID unique TID
stack		unique
signals		shared*
memory file descriptors mutexes ...	unique[1]	shared

[1] copied on exec

Inter-process communication (IPC)

- signals
  - no data, hard to use
- shared memory
  - only data, no events / change notification
  - needs synchronization
- (POSIX | System-V) message queues
- pipes / FIFOs
  - one-to-one, unidirectional
- sockets (local, network)
  - one-to-many or many-to-many

Signals

- Sending

```
int kill(pid_t pid, int sig)
```

- Receiving

```
void handler1(int sig);
void handler2(int sig, siginfo_t *siginfo, void *contex);
void sigaction(int signum, const struct sigaction *new,
               struct sigaction *old);
```

- Blocking

```
int sigprocmask(int how, const sigset_t *set, sigset_t*oldset);
int pthread_sigmask(...);
```

- Default action for some signals is to terminate the receiving process!

```
man 7 signal
```

Shared memory

```
int open(const char *pathname, int flags, mode_t mode);
int shm_open(const char *name, int oflag, mode_t mode);
int shm_unlink(const char *name);

void *mmap(void *addr, size_t length, int prot, int flags,
           int fd, off_t offset);
int munmap(void *addr, size_t length);
```

- use `shm_open` to open an existing or create a new shared memory object identified by `name`
- use `mmap` to map it into process memory space
- alternatively use `mmap` on regular files openned by `open` to get "persistent" filesystem backed shared memory

`man 7 shm_overview`