3.10 Einschlägige Systemaufrufe bei SunOS (Auswahl)

3.10.1 Prozesse

- Erzeugung, Überlagerung

  create a new process
  ```
  pid_t fork()
  ```
  execute a file
  ```
  int execve(char *path, char *argv[], char *envp[])
  ```

- Prozeß beenden (und tilgen)

  terminate a process
  ```
  void _exit(int status)
  ```

- Prozeßidentifikator

  get process identification
  ```
  int getpid()
  ```
  get process identification of parent
  ```
  int getppid()
  ```

- Warten auf Beendigung eines untergeordneten Prozesses

  wait for process to terminate or stop, examine returned status
  ```
  pid_t wait(int *statusp)
  ```

3.10.2 Semaphore

- Erzeugung

get set of semaphores

```
int semget(key_t key, int nsems, int semflg)
```

- Operationen

semaphore operations

```
int semop(int semid, struct sembuf *sops, int nsops)
   struct sembuf {
      short sem_num;     /* semaphore # */
      short sem_op;      /* semaphore operation */
      short sem_flg;     /* operation flags */
   };
```

semaphore control operations

```
int semctl(int semid, int semnum, int cmd, semun arg)
   union semun {
      int              val;
      struct semid_ds  *buf;
      ushort           *array;
   };
   #define  GETNCNT  3        /* get semncnt */
   #define  GETPID   4        /* get sempid */
   #define  GETVAL   5        /* get semval */
   #define  GETALL   6        /* get all semval's */
   #define  GETZCNT  7        /* get semzcnt */
   #define  SETVAL   8        /* set semval */
   #define  SETALL   9        /* set all semval's */
```

3.10.3 Nachrichten

3.10.3.1 Kommunikationskanäle, Pipes

- Erzeugung

  create an interprocess communication channel

  ```
  int pipe(int fd[2])
  ```

- Operationen

  read input

  ```
  int read(int fd, char *buf,  int nbyte)
  ```

  read input and scatter

  ```
  int readv(int fd, struct iovec *iov, int iovcnt)
  ```

  write output

  ```
  int write(int fd, char *buf,  int nbyte)
  ```

  write output and gather

  ```
  int writev(int fd, struct iovec *iov, int iovcnt)
  ```

3.10.3.2 Briefkästen, mailboxes, queues, ...

- Erzeugung

get message queue

```
int msgget(key_t key, int msgflg)
```

- Operationen

message operations

```
int msgsnd(int msqid,
           struct msgbuf *msgp, int msgsz,
           int msgflg)
int msgrcv(int msqid,
           struct msgbuf *msgp, int msgsz,
           long msgtyp, int msgflg)
```

3.10.3.3 Kommunikationsendpunkte, Sockets

- Erzeugung

create an endpoint for communication

```
int socket(int domain, int type, int protocol)
```

create a pair of connected sockets

```
int socketpair(int d, int type,
               int protocol, int sv[2])
```

- Operationen

bind a name to a socket

```
int bind(int s, struct sockaddr *name, int namelen)
```

initiate a connection on a socket

```
int connect(int s,
           struct sockaddr *name, int namelen)
```

send a message from a socket

```
int send(int s, char *msg, int len, int flags)
int sendto(int s, char *msg, int len, int flags,
           struct sockaddr *to, int tolen)
int sendmsg(int s, struct msghdr *msg, int flags)
```

receive a message from a socket

```
int recv(int s, char *buf, int len, int flags)
int recvfrom(int s, char *buf, int len, int flags,
            struct sockaddr *from, int *fromlen)
int recvmsg(int s, struct msghdr *msg, int flags)
```

listen for connections on a socket

```
int listen(int s, int backlog)
```

accept a connection on a socket

```
int accept(int s, struct sockaddr *addr,
          int *addrlen)
```