Semaphores

- Theory
- C functions
- Example

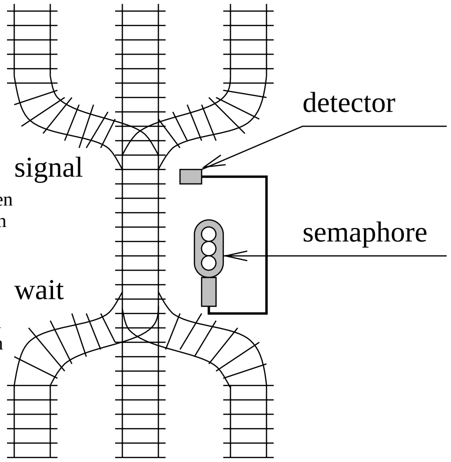
Railiway example

V signal

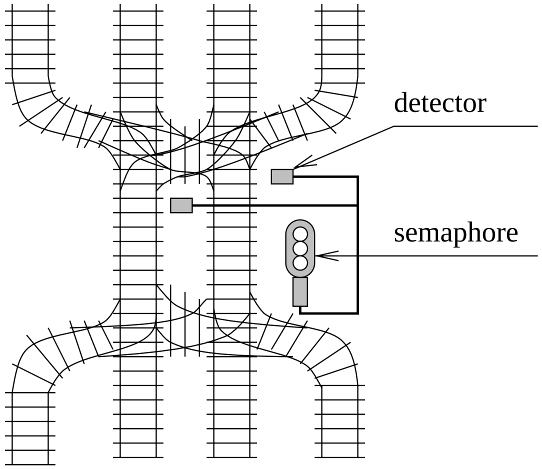
vrijmaken
verhogen

P wait

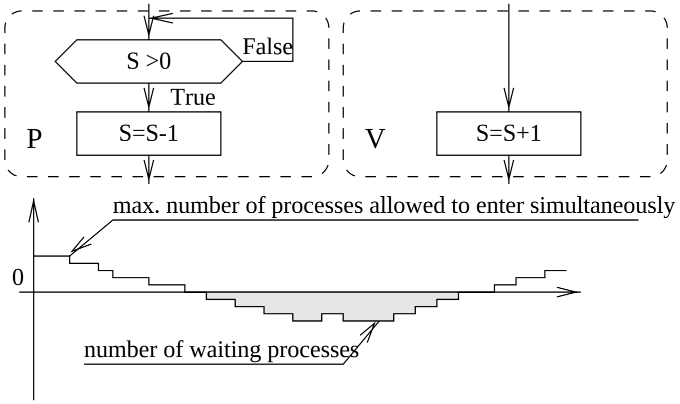
passeren
proberen



Another railiway example



Semaphore implementation


S >0
False
True
P
S=S-1
V
S=S+1
max. number of processes allowed to enter simultaneously
0
number of waiting processes

Functions

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

Get array of **nsems** semaphores.

```
int semop(int semid, struct sembuf *sops, size_t nsops);
```

Perform semaphore operations specified by **sops**.

```
int semctl(int semid, int semnum, int cmd,
                    /* union semun arg */ ...);
```

Perform semaphore control operations specified by **cmd**.

GTEVAL – get value of the semaphore
SETVAL – set value of the semaphore

```
#include <sys/types.h>
#include<sys/ipc.h>
#include <sys/sem.h>

static int sem;
static struct sembuf buf[1];

void declare( void )
{
sem = semget( 1, 1, IPC_CREAT | 0666 );
buf[0].sem_num = 0;
buf[0].sem_flg = 0;
}

void init( int i )
```

```
{
semctl( sem, 0, SETVAL, i );
}
```

```
void sem_wait( void )
{
buf[0].sem_op = -1;
semop( sem, buf, 1 );
}

void sem_signal( void )
{
buf[0].sem_op = 1;
semop( sem, buf, 1 );
}

void delay( void )
{
int i, j;
for (i=0; i<100000; i++)
  j=i+1;
}

void sem_print( char *msg, int proc )
{
sem_wait();
printf( "Message %d\n", proc );
delay();
printf( "            from: %s\n", msg );
sem_signal();
delay();
}
```

```
int main( int narg, char *argv[] )
{
int i;
int pid;

pid=fork();
if (pid == 0)
  {
  for (i=0; i<5; i++)
    sem_print( "Child", i );
  }
else if (pid > 0)
  {
  for (i=0; i<5; i++)
    sem_print( "Parent", i );
```

```
  }
else
  printf( "Error\n" );
}
```

```
Message 0                Message 0
            from: Child              from: Child
Message 0                Message 1
            from: Parent             from: Child
Message 1                Message 2
            from: Parent             from: Child
Message 2                Message 0
            from: Parent             from: Parent
Message 1                Message 3
            from: Child              from: Child
Message 2                Message 1
            from: Child              from: Parent
Message 3                Message 2
Message 3                            from: Parent
            from: Parent Message 4
Message 4                            from: Child
            from: Parent Message 3
            from: Child              from: Parent
Message 4                Message 4
```