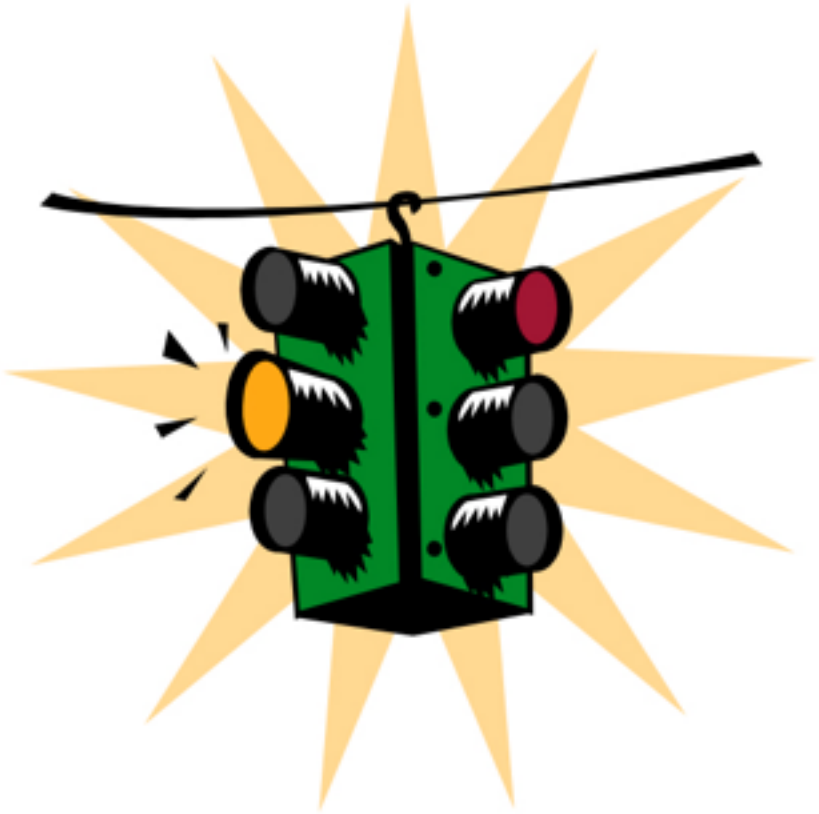
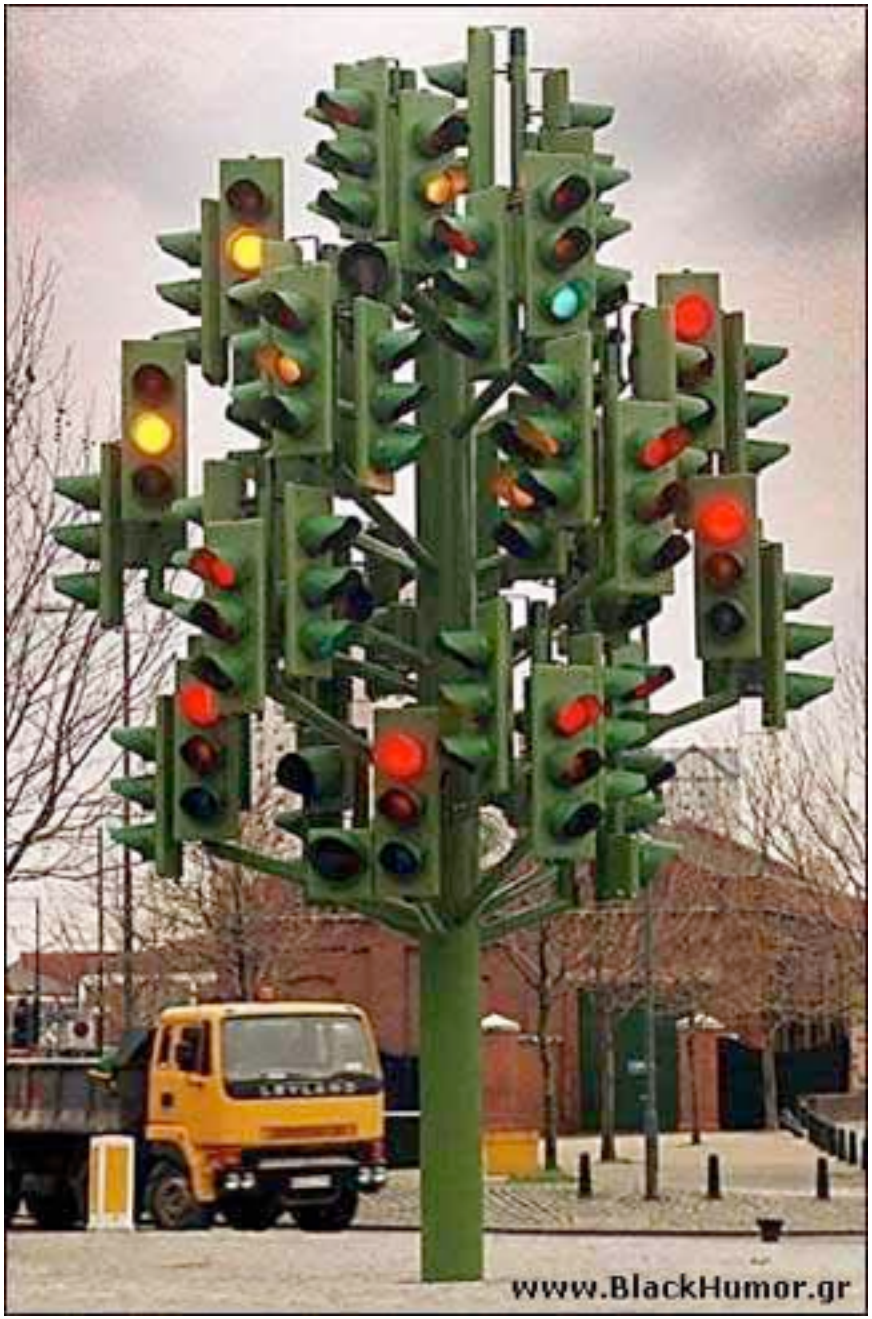
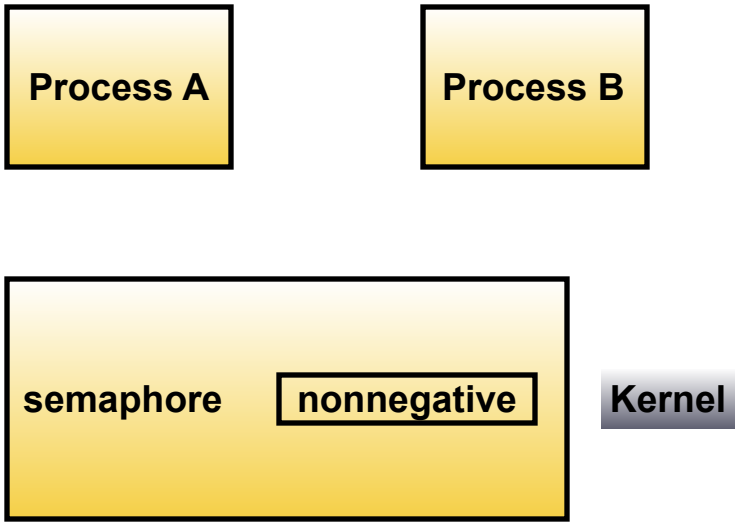


Semaphores

semaphores



- Semaphores
  - Synchronization
  - Avoid starvation
  - Avoid deadlock
  - Atomic
  - Integer value
    - Resource counter



synchronize operations on non-shareable resources

- atomic operation
- avoids busy wait
- two operations
  - P & V
    - on non-negative variables
      - semaphores

Edsger W. Dijkstra
1930–2002

Dijkstra, E. W. (March 1968). "Letters to the editor: go to statement considered harmful". *Communications of the ACM* **11** (3): 147–148. doi:10.1145/362929.362947. ISSN 0001-0782.

- P & V
  - only valid operations on a semaphore variable
  - operations on same semaphore are mutually exclusive
    - sequential effect in no particular order
    - any waiting process may be selected

```
P(Sem):
    -if possible decrements  Sem  by  1
    -else process must wait until Sem > 0

V(Sem):
    -if there is a process waiting -> release
        -else increment  Sem  by  1
```

- Behavior

```
syscall P(int Sem)
{
  if ( Sem > 0 )  Sem--;
   else  block on Sem;
}

syscall V(int Sem)
{
  if (process waiting on Sem) wake_up a process;
   else Sem++;
```

```
process p1 ()
{
  while(true)
    {
        non critical code;
        P(Sem);
        Critical Section;
        V(Sem);
        non critical code;
    }
}
process p2 ()
{
  while(true)
    {
        non critical code;
        P(Sem);
        Critical Section;
        V(Sem);
        non critical code;
    }
}
```

- Binary semaphores

- may only have values of 0 or 1
  - 1 semaphore NOT BUSY
  - 0 semaphore IS BUSY

Counting semaphores

- Make take any integer value
  - **sem > 0**
    - number of processes that may still continue execution after a `P(sem)` operation
  - **sem == 0**
    - no more processes may continue after the execution of a `P(sem)` operations
  - **sem < 0**
    - number of processes waiting on semaphore `sem`

- Counting semaphores

```
P(int Sem)
{
   Sem--;
   if (Sem < 0 ) block on Sem;
}
```

```
V(int Sem)
{
     Sem++;
     if ( Sem <= 0 ) wake up a process;
}
```

- System V implementation
  - A *set* of nonnegative integer
    - System defined maximum
  - Maintained by the kernel
- Creation
  - unique data structure
  - created/maintained by kernel
  - implemented as an integer
    - Stored in kernel space
    - Accessed through system calls
    - ***Atomic***

- **For every set of semaphores kernel maintains following structure**

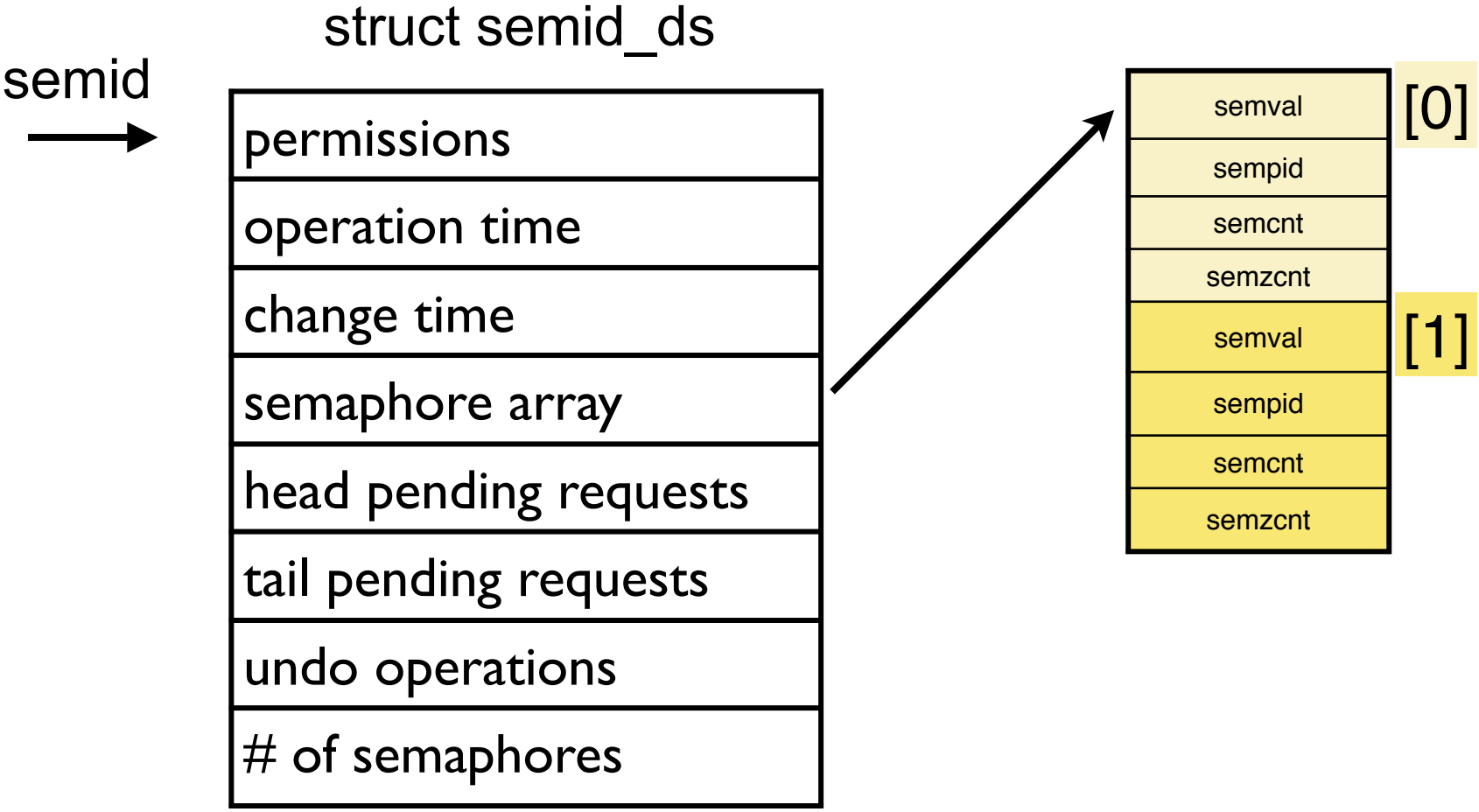
```
#include <sys/types.h>
#include <sys/ipc.h>                    /* defines ipc_perm structure*/
struct semid_ds {
    struct  ipc_perm    sem_perm;     /* operations permission   */
    __time_t            sem_otime;    /* time of last semop      */
    __time_t            sem_ctime;    /* time of last change     */
    struct  sem         *sem_base;    /* ptr to 1st semaphore in set */
    struct  sem_queue   *sem_pending;/* pending operations*/
    struct  sem_queue   *sem_pending_last; /* last pending operation*/
    struct  sem_undo    *undo;        /* undo requests on this set  */
    unsigned long int   sem_nsems;    /* # of semaphores in set  */

};
```

Each member is described by following structure

```
struct sem  {
    ushort semval;       /* semaphore value, nonnegative   */
    short  sempid;       /* pid of last operation          */
    ushort semncnt;      /* # awaiting semval > cval       */
    ushort semzcnt;      /* # awaiting semval = 0          */
};
```

- **Kernel data structure for a semaphore set**




```
struct     sem  {
      ushort     semval;    /* semaphore value, nonnegative */
      short      sempid;    /* pid of last operation        */
      ushort     semncnt;   /* # awaiting semval > cval      */
      ushort     semzcnt;   /* # awaiting semval = 0         */
};
```

- **Kernel data structure for a semaphore set**
  - `sem_pending`
    - Linked list of pending semaphore operations
  - `sem_pending_last`
    - end of list
  - `sem_undo`
    - undo requested semaphore operation
    - process exits without releasing semaphore operation
    - used with `SEM_UNDO` flag
    - helps reduce deadlock possibilities

- Semaphore creation/access

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

int semget (key_t key,  int nsems,  int semflag);

returns semid

nsems: number of semaphores (creation only)

semflag:
```

Numeric	Symbolic	Description
0400	SEM_R	Read by owner
0200	SEM_A	Alter by owner
0040	SEM_R >> 3	Read by group
0020	SEM_A >> 3	Alter by group
0004	SEM_R >> 6	Read by world
0002	SEM_A >> 6	Alter by world
	IPC_CREAT	
	IPC_EXCL	

- `semget`
  - `key`
    - to generate unique semaphore id
    - IPC_PRIVATE
  - `nsems`
    - number of semaphores
    - only used for creation
  - semflag
    - permission
    - IPC_CREAT

      semaphore array is not initialized
    - IPC_EXCL

```
/* Creating sets of semaphores */
#include <iostream>
#include <cstdio>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/sem.h>
using namespace std;
int
main( ){
  int       sem1, sem2, sem3;
  key_t     ipc_key;
  ipc_key = ftok(".", 'S');
  if ((sem1 = semget(ipc_key, 3, IPC_CREAT | 0660)) == -1) {
    perror("semget: IPC_CREAT | 0660");
  }
  printf("sem1 identifier %d\n", sem1);
  if ((sem2 = semget(ipc_key, 3, IPC_CREAT|IPC_EXCL|0600)) == -1) {
    perror("semget: IPC_CREAT | IPC_EXCL | 0660");
  }
  printf("sem2 identifier %d\n", sem12);
  if ((sem3 = semget(IPC_PRIVATE, 3, 0600)) == -1) {
    perror("semget: IPC_PRIVATE");
  }
  printf("sem3 identifier %d\n", sem3);
  return 0;
}
```

semaphores


```
$./a.out
sem1 identifier 2818048
semget: IPC_CREAT | IPC_EXCL | 0600: File exists
sem2 identifier -1
sem3 identifier 65537

$ ./a.out
sem1 identifier 2818048
semget: IPC_CREAT | IPC_EXCL | 0600: File exists
sem2 identifier -1
sem3 identifier 65538
```

```
$ ipcs -s
T     ID      KEY         MODE        OWNER    GROUP
Semaphores:
s 2818048 0x530253c3 --ra-ra----   dcanas    staff
s   65537 0x00000000 --ra-------   dcanas    staff
s   65538 0x00000000 --ra-------   dcanas    staff
```

- Semaphore control

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

int semctl (int semid,  int semnum, int cmd, union semnum arg);
```

use cmd :

IPC_RMID	to remove semaphore
IPC_STAT	returns current value in arg
IPC_SET	to set semaphore value - initialization set semaphore to arg.val

```
union semnun {
    int     val;              /* used for SETVAL only          */
    struct semid_ds  *buf;    /* used for IPC_STAT and IPC_SET */
    ushort int *array         /* used for GETALL & SETALL      */
   struct seminfo   *_buf;    /* buffer for IPC_INFO           */
} arg;
```

- arg values
  - an integer used with SETVAL to indicate a specific values for a particular semaphore
  - a reference to `semid_ds` structure for information returned by IPC_STAT or IPC_SET
  - a reference to an array of type unsigned integers used to either initialize the set or as a return location for GETALL
  - a reference to a `seminfo` structure when IPC_INFO is requested

- Semaphore Control (cmd values)
  - on specific set
    - IPC_STAT
      - values of sem_id structure returned in fourth argument
    - IPC_SET
      - permissions, uid, gid information
      - as specified in fourth argument
    - IPC_RMID
      - remove
  - on entire set
    - GETALL
      - returns values of the semaphore set
    - SETALL
      - sets values of the semaphore set

- Semaphore Control
  - on individual semaphores
    - GETVAL
      - returns value of individual semaphore referenced by `semnun`
    - SETVAL
      - sets value of individual semaphore referenced by `semnun`
    - GTPID
    - GETCNT
      - returns number of processes waiting on semaphore to increase value
    - GETZCNT
      - returns number of processes waiting on semaphore to become 0

- Semaphore Operations

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

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

pointer sops points to an array of the following structure:

```
struct sembuf {
    unsigned short   sem_num;      /* semaphore #          */
    short int        sem_op;       /* semaphore operation  */
    short  int       sem_flag;     /* operations flag      */
};
```

nsops : number of elements in array of sembuf

Each element specifies an operation for one particular semaphore value specified by sem_num

- semaphore control
  - sem_flag
    - IPC_NOWAIT
      - return immediately if operation cannot be performed
      - no other semaphore sets are modified
    - SEM_UNDO
      - if IPC_NOWAIT not specified
        - undo operations if a blocked operation subsequently fails

- Semaphore Operations

1. If `sem_op` is positive, the value of `sem_op` is added to the semaphore's current value. This corresponds to the release of resources that a semaphore controls.

2. If `sem_op` is zero, the caller wants to wait until the semaphore's value becomes zero.

3. If `sem_op` is negative, the caller wants to wait until the semaphore's value becomes greater or equal to the absolute value of `sem_op`. Then the absolute value of `sem_op` is subtracted from the semaphore's current value. This corresponds to the allocation of resources

- actions when `semop` is positive

flag set	action taken by `semop`
	add `sem_op` to `semval`
`SEM_UNDO`	add `sem_op` to `semval` and update the undo counter for the semaphore

- actions when `semop` is zero

condition	flag set	action taken by `semop`
`semval` == 0		Return immediately
`semval` != 0	`IPC_NOWAIT`	Return -1 immediately and set `errno` to `EAGAIN`
`semval` != 0		Increment `semzcnt` for the semaphore and wait until: -`semval` == 0, then adjust `semzcnt` and return -`semid` is removed, then return -1 and set `errno` to EIDRM -a signal is caught, then adjust `semzcnt` and set `errno` to `EINTR`

- actions when semop is negative

condition	flag set	action taken by semop
semval >= abs(semop)		Subtract abs(sem_op) from semval
semval >= abs(semop)	SEM_UNDO	Subtract abs(sem_op) from semval and update the undo counter for the semaphore
semval < abs(semop)		increment semcnt for the semaphore and wait until -semval >= abs(Semop), then adjust semcnt and subtract as noted in the previous two rows of table -semid is removed, then return -1 and set errno to EIDRM -a signal is caught, then adjust semcnt and set errno to EINTR
semval < abs(semop)	IPC_NOWAIT	return -1 immediately and set errno to EAGAIN

- operations on a set of semaphores