I33 Application Data Sharing with z/OS Shared Memory

Detlef Dewitz
DEFObonn GmbH, Bonn, Germany

Introduction

- Preliminary notes
  - All informations and samples are based on z/OS 1.7
  - z/OS 1.9 has some new functions and parameters which are not discussed in this session
  - This session offers an introduction to the subject of application data sharing. Hence, not all details will be discussed.

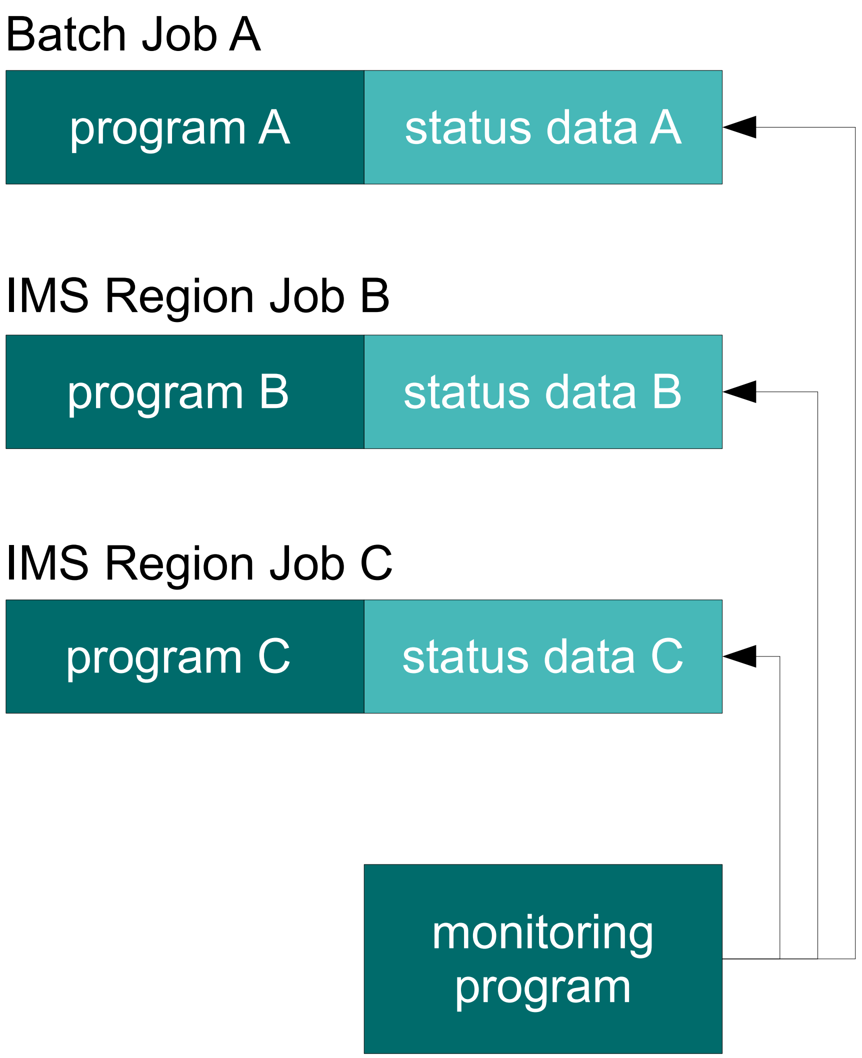
DEFObonn

1. Needs

What may be your needs ?

DEFObonn

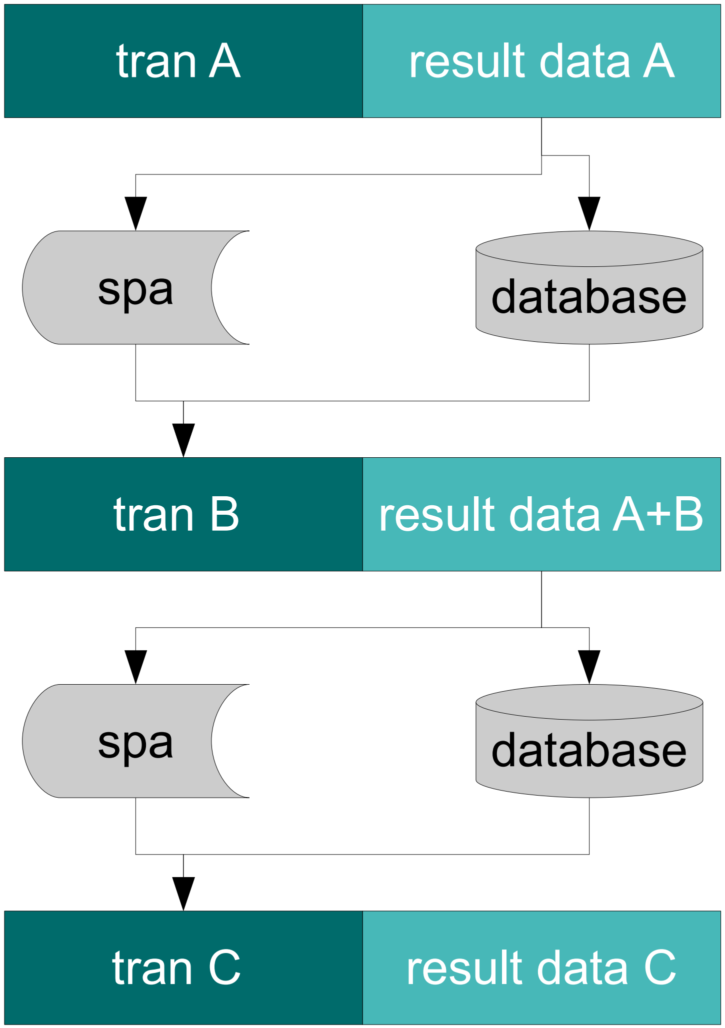
1.1 Case 1 – Live Monitoring



- A live monitoring program shall keep track of the processing status of your applications
- This can be i.e.
  - TSO/ISPF applications
  - Batch Jobs
  - IMS Jobs
  - Web applications
  - ...
- You cannot use online databases!

DEFObonn

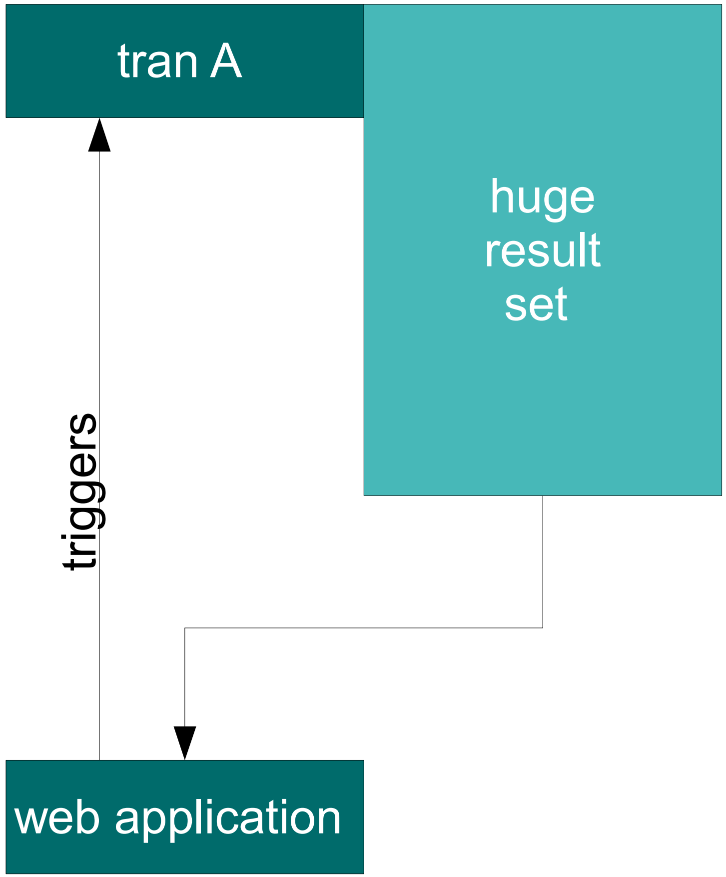
1.2 Case 2 – IMS Conversational Workflow



- A workflow of conversational IMS transactions, i.e. results of tran A are needed by the 2nd following tran C
  - Results are to be buffered in the SPA (or database, if data size > SPA size)
  - Tran B – in the middle – has to process data in SPA although not needed
  - Large data sizes cost a lot of MIPS for compression and database i/o

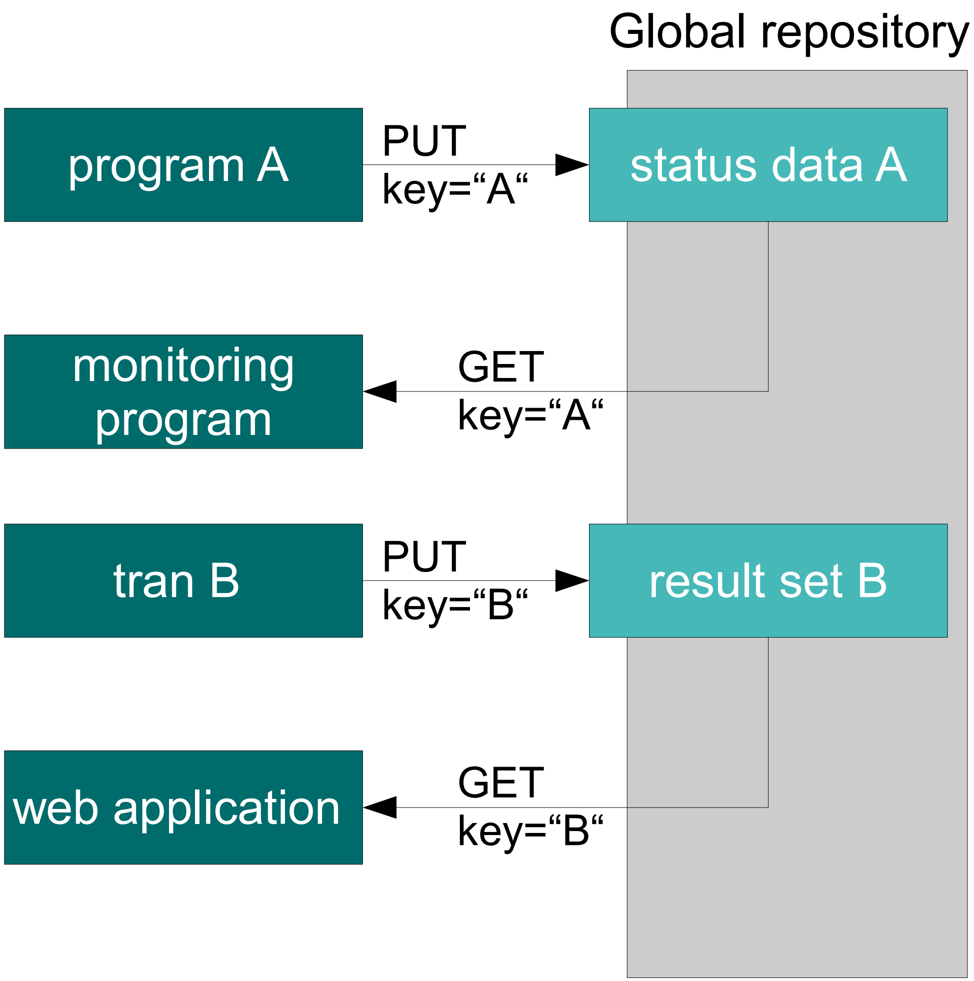
DEFObonn

1.3 Case 3 – Web Application with a huge Result Set



- A web application triggers a IMS transaction
- The IMS transaction creates a huge result set
- This result set shall be displayed by the web application now or later

1.4 Case 4 – A Global Repository for Temporary Data



- Instead of single solutions for each application think also about ...
- ... a global repository for all your needs which
  - fits all sizes of data
  - is key based
  - uses heap based algorithms
  - includes garbage collection
  - includes time based expirations

1.5 Case 5 – IMS Database Cache

What have we done in our project ?

- One of the project's databases contains images of cobol working storage sections for read only purposes
- IMS reporting showed that this database
  - was 2nd off most used IMS databases in the daytime
  - GU/GN were 99,9 % of the IMS calls
  - was used by a lot of transactions (MPP and BMP)

DEFObonn

1.5.1 Case 5 – IMS Database Cache

Preconditions

- All IMS calls of the applications are encapsulated in one global sub-module.
- This global sub-module is written in assembler
- Updates to this database occur once or twice per week
- Database contains different versions of the working storage images
- One version of the image contains data of about 40 MB
- 1-2 IMS systems per LPAR in production
- 1-7 IMS systems per LPAR in development

DEFObonn

1.5.2 Case 5 – IMS Database Cache

Objectives

- Decrease of MIPS
- Decrease of IMS calls
- Caching of the working storage images in main storage
- Only one cache per image per IMS system
- No changes in the applications except for the global sub-module
- Caches must be manageable in production and development

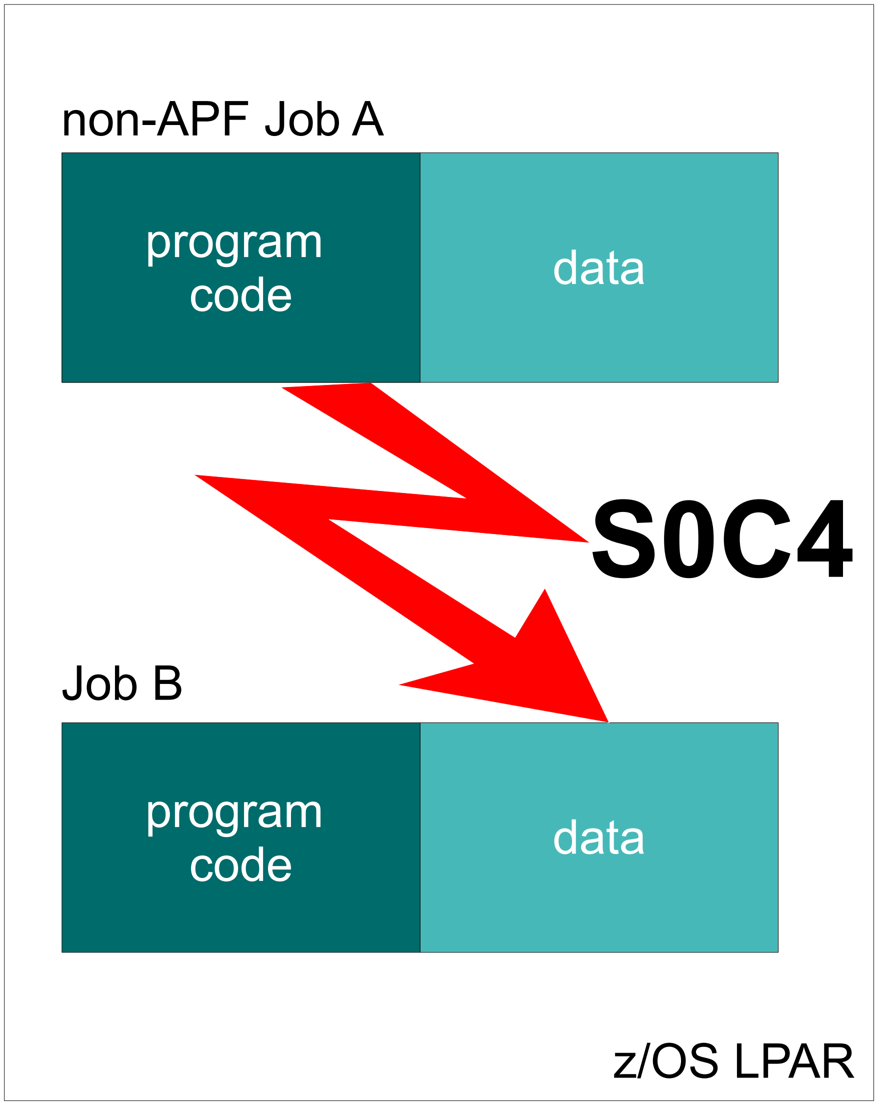
DEFObonn

2. Problems

What are the problems ?

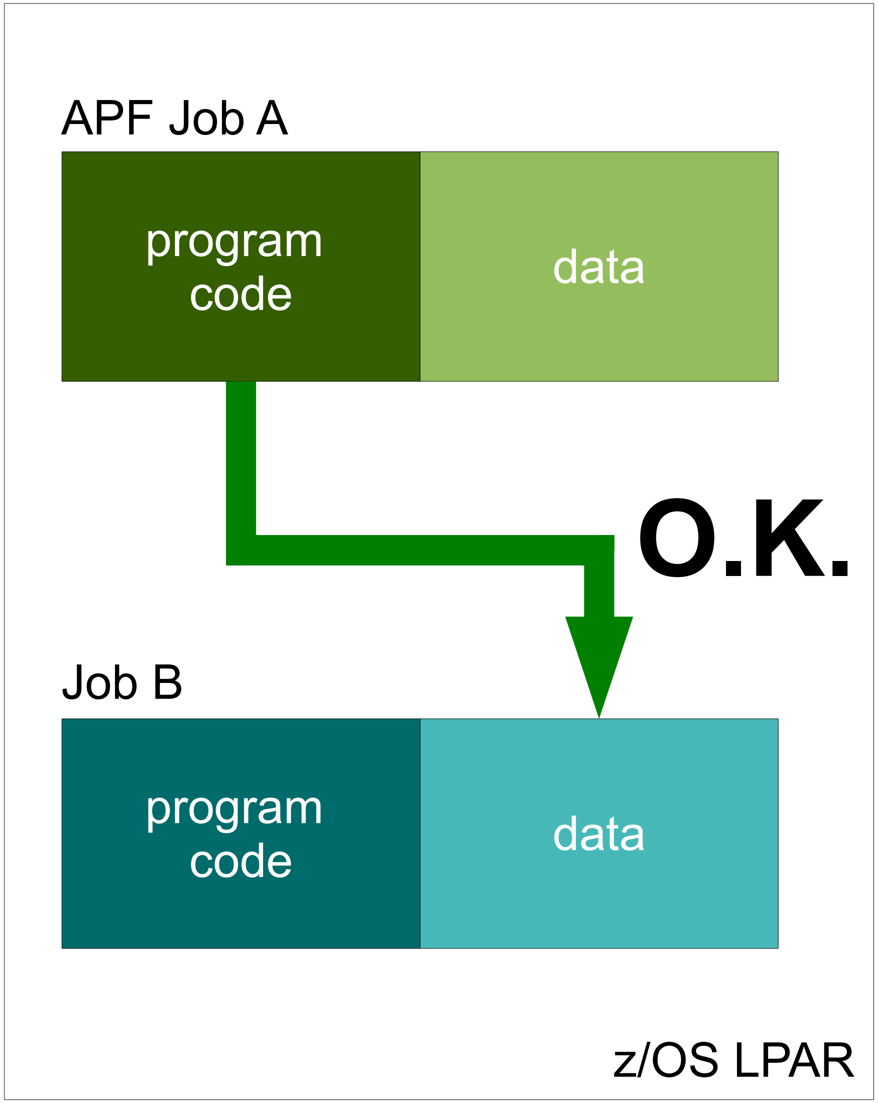
DEFObonn

2.1 Problem S0C4



- A non APF authorized program is not allowed to read or write data outside its own address space
- If anyhow a program tries to access another address space, the program will be interrupted with system abend 0C4

2.2 Problem APF Authorization



- APF authorization of the program will master the S0C4 problem, but ...
- Programming is tricky ...
- All libraries of your steplib/joblib concatenation have to be authorized ...
- Your ITO will usually not allow authorizing your standard application

DEFObonn

3. IPC concept

The Inter-Process Communication
(short IPC)

DEFObonn

3.1 IPC Definition of wikipedia.org

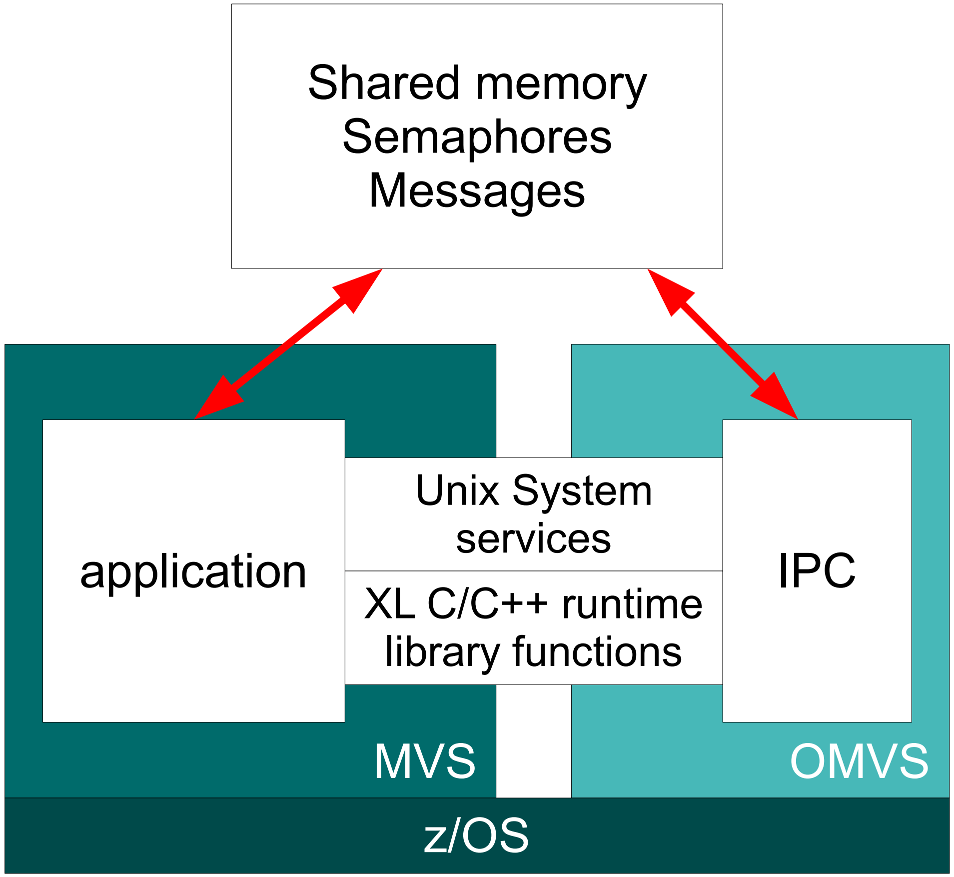
- Inter-Process Communication (IPC) is a set of techniques for the exchange of data among multiple threads in one or more processes....
- Processes may be running on one or more computers connected by a network....
- IPC techniques are divided into methods for message passing, synchronization, shared memory, and remote procedure calls (RPC)....
- The method of IPC used may vary based on the bandwidth and latency of communication between the threads, and the type of data being communicated....

3.2 IPC on z/OS

- IPC is part of OMVS (POSIX compatible UNIX)
- Processes (TSO-sessions, Jobs, IMS-TRX) of one LPAR
- IPC techniques are divided into methods for
  - shared memory
  - synchronization
  - message passing (not dicussed in this presentation)
- Remote procedure calls (RPC) are not yet supported

DEFObonn

3.3 IPC diagram



- IPC has rights of user 'root', so no extra APF authorization is neccessary
- IPC methods can called from the MVS address room using
  - XL C/C++ runtime library functions
  - Assembler unix system services

DEFObonn

How to control IPC

4.1 Clist BPX

A little clist helps ...

<userid>.user.clist(bpx)

```
/* REXX */
PARSE ARG COMMAND

"ALLOC DD(STDIN) DUMMY REUSE"
"ALLOC DD(STDOUT) DA(*) REUSE"
"ALLOC DD(STDERR) DA(*) REUSE"

EXITRC = BPXWUNIX(COMMAND,'DD:STDIN','DD:STDOUT','DD:STDERR','0')

EXIT(EXITRC)
```

ATTENTION:
You cannot use this clist at any ISPF screen, because OMVS commands are case sensitive

DEFObonn

4.1.1 Clist BPX - Example

... entering a OMVS command from TSO/ISPF 6.

```
                                  ISPF Command Shell
Enter TSO or Workstation commands below:

 ===> %bpx nslookup www.ims-society.org______________________
```

```
Server:  localhost
Address:  127.0.0.1
  Non-authoritative answer:
Name:    www.ims-society.org
Address:  192.67.198.56
Defaulting to nslookup version 4
Starting nslookup version 4
***
```

4.1.2 Clist BPXSPOOL

In a batch job try this ...

<userid>.user.clist(bpxspool)

```
/* REXX */
PARSE ARG COMMAND
EXITRC = BPXWUNIX(COMMAND,'DD:STDIN','DD:STDOUT','DD:STDERR','0')
EXIT(EXITRC)
```

Jobcards

```
//UNIX    EXEC PGM=IKJEFT01,PARM='BPXSPOOL ps -Aj'
//SYSEXEC  DD DISP=SHR,DSN=<USERID>.USER.CLIST
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD DUMMY
//STDOUT   DD SYSOUT=*
//STDERR   DD SYSOUT=*
//STDIN    DD DUMMY
```

4.2 OMVS command ipcs

- The command `ipcs` retrieves status information about active IPC objects

`ipcs -a`

- Shows all active message queue, shared memory and semaphore objects

`ipcs -q`

- Shows all active message queues

`ipcs -m`

- Shows all active shared memory segments

`ipcs -s`

- Shows all active semaphore sets

`ipcs -w`

- Shows message queue wait status and semphore adjustment status

`man ipcs`

- shows help description for command `ipcs`

4.3 OMVS command ipcrm

- The command `ipcrm` removes an IPC objects

  `ipcrm -q` *`msgid`*

  `ipcrm -Q` *`msgkey`*

  - Removes an active message queue with the associated *`msgid`* or *`msgkey`*

  `ipcrm -m` *`shmid`*

  `ipcrm -M` *`shmkey`*

  - Removes an active shared memory segment with the associated *`shmid`* or *`shmkey`*

  `ipcrm -s` *`semid`*

  `ipcrm -S` *`semkey`*

  - Removes an a active semaphore set with the associated *`semid`* or *`semkey`*

  `man ipcrm`

  - Shows help description for command `ipcrm`

4.4 OMVS command ps

- The command `ps` returns the status of an OMVS process

  `ps -A`

  `ps -Aj`

  - Shows all active processes associated with an OMVS process id

  `man ps`

  - Shows help description for command `ps`

- If you are looking for a specific process you can filter the output using grep, i.e.

  `ps -A | grep 4711`

  - where `4711` is the process id you are looking for

DEFObonn

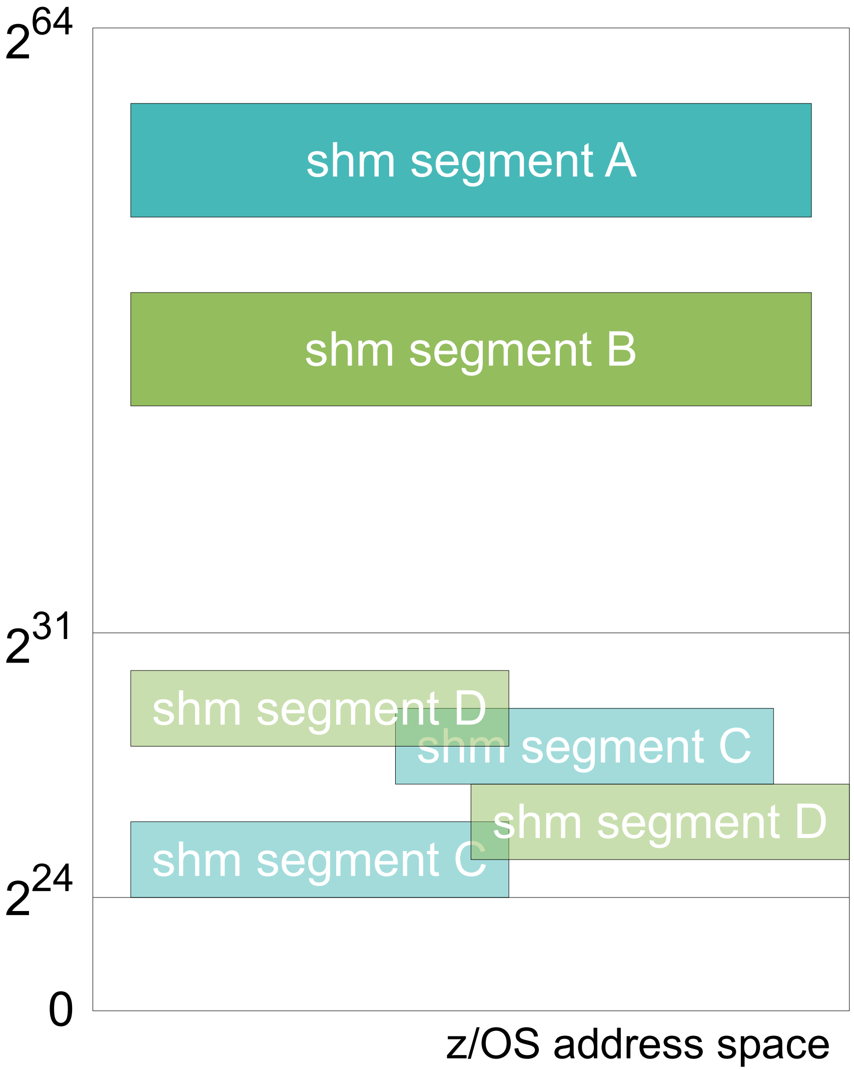
5. IPC Shared Memory

IPC Shared Memory

5.1 IPC Shared Memory Diagram

Shared memory segments

- 64 bit address space:
  - Multiple of 1 gigabyte segments
  - The address is fixed to all
- 31 bit address space:
  - Sizes < 1 megabyte
  - Multiple of 1 megabyte segments
  - The address is a virtual



DEFObonn

5.2 IPC SHM methods

- *shmget* - Get a Shared Memory Segment
  - Creates a new shm segment and returns it's ID
  - Gets the ID of an existing segment
- *shmat* - Shared Memory Attach Operation
  - Attaches the shared memory segment associated with the ID to the application address space
- *shmdt* - Shared Memory Detach Operation
  - Detaches from the applications address space the shared memory segment located at the given address
- *shmctl* - Shared Memory Control Operations
  - Obtains status information
  - Changes permissions
  - Removes a shared memory segment

5.3 IPC SHM steps

- Steps

1. Define an identifying 4 byte key

   shmkey=0x00004711

2. Create a shared memory

   shmid = semget (shmkey, size, IPC_CREAT)

3. Attach the shared memory to your address space

   buffAddr = shmat (shmid)

5. Do your work

6. Detach the shared memory from your address space

   rc = shmdt (buffAddr)

7. Delete the shared memory segment permanently

   rc = shmctl (shmid, IPC_RMID)

5.3.1 SHM sample program EMEA0001 page 1

EMEA0001.C

```
#define _XOPEN_SOURCE
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <sys/ipc.h>
#include <sys/shm.h>

#define SHM_KEY         0x00004711
#define SHM_MODE        0777
#define BUFFER_SIZE     1023

typedef struct buffer_t {
  int counter;
  int data [BUFFER_SIZE];    /* The buffer which holds the data. */
} buffer_t;

unsigned int sharekey = SHM_KEY;
unsigned int size = sizeof (struct buffer_t);
```

5.3.2 SHM sample program EMEA0001 page 2

```
int main (int argc, *argv []) {

  buffer_t      *buffer;
  int            shmid, i, rc;

  system ("%bpx ipcs -a");
  /* Create SHM segment */
  shmid = shmget (sharekey, size, IPC_CREAT | SHM_MODE);
  if (shmid == -1) {
    perror ("shmget");
    return;
  }
  system ("%bpx ipcs -a");

  /* Attach SHM segment to own address space */
  buffer = (buffer_t *) shmat (shmid, (char *) NULL, 0);
  if (buffer == (buffer_t *) -1) {
    perror ("shmat");
    return;
  }
  system ("%bpx ipcs -a");
```


DEFObonn

5.3.3 SHM sample program EMEA0001 page 3

```
  /* Initialize our application data in shared memory */
  for (i = 0; i < BUFFER_SIZE; i ++) {
    (*buffer).data [i] = i;
  }
  (*buffer).counter = -1;

  /* Detach the shared memory segment form our address space */
  rc = shmdt ((char *) buffer);
  if (rc  == -1) {
    perror ("shmdt");
    return;
  }
  system ("%bpx ipcs -a");

  /* Delete permanently the shared memory segment */
  rc = shmctl (shmid, IPC_RMID, NULL);
  if (rc == -1) {
    perror ("shmctl");
  }
  system ("%bpx ipcs -a");
} /* main */
```


DEFObonn

6. IPC Semaphores

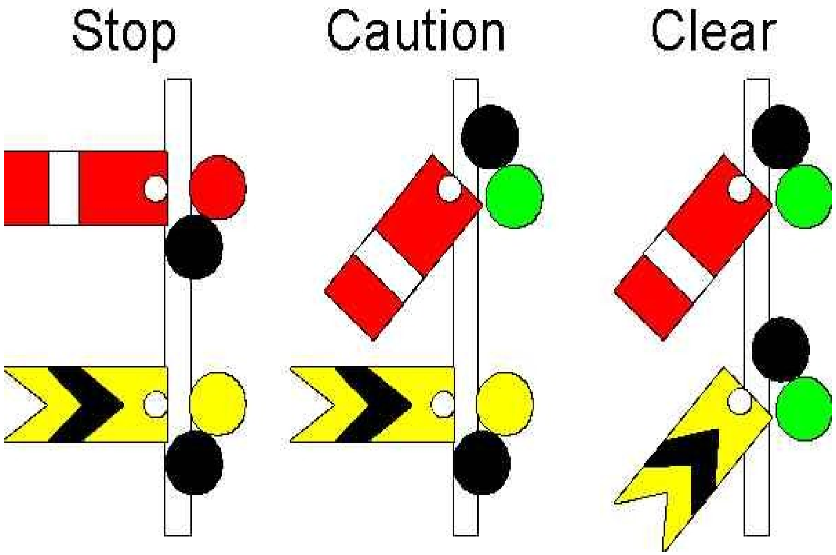
Locking strategies using IPC semaphores

6.1 What is a Semaphore ?

One of the earliest forms of fixed railway signal is the semaphore

6.2 Semaphore Definition of wikipedia.org

- A semaphore, in computer science, is a protected variable (an entity storing a value) or abstract data type (an entity grouping several variables that may or may not be numerical) which constitutes the classic method for restricting access to shared resources, such as shared memory, in a multiprogramming environment ...
- ... A counting semaphore is a counter for a set of available resources, rather than a locked/unlocked flag of a single resource...
- ... It was invented by Edsger Dijkstra ...



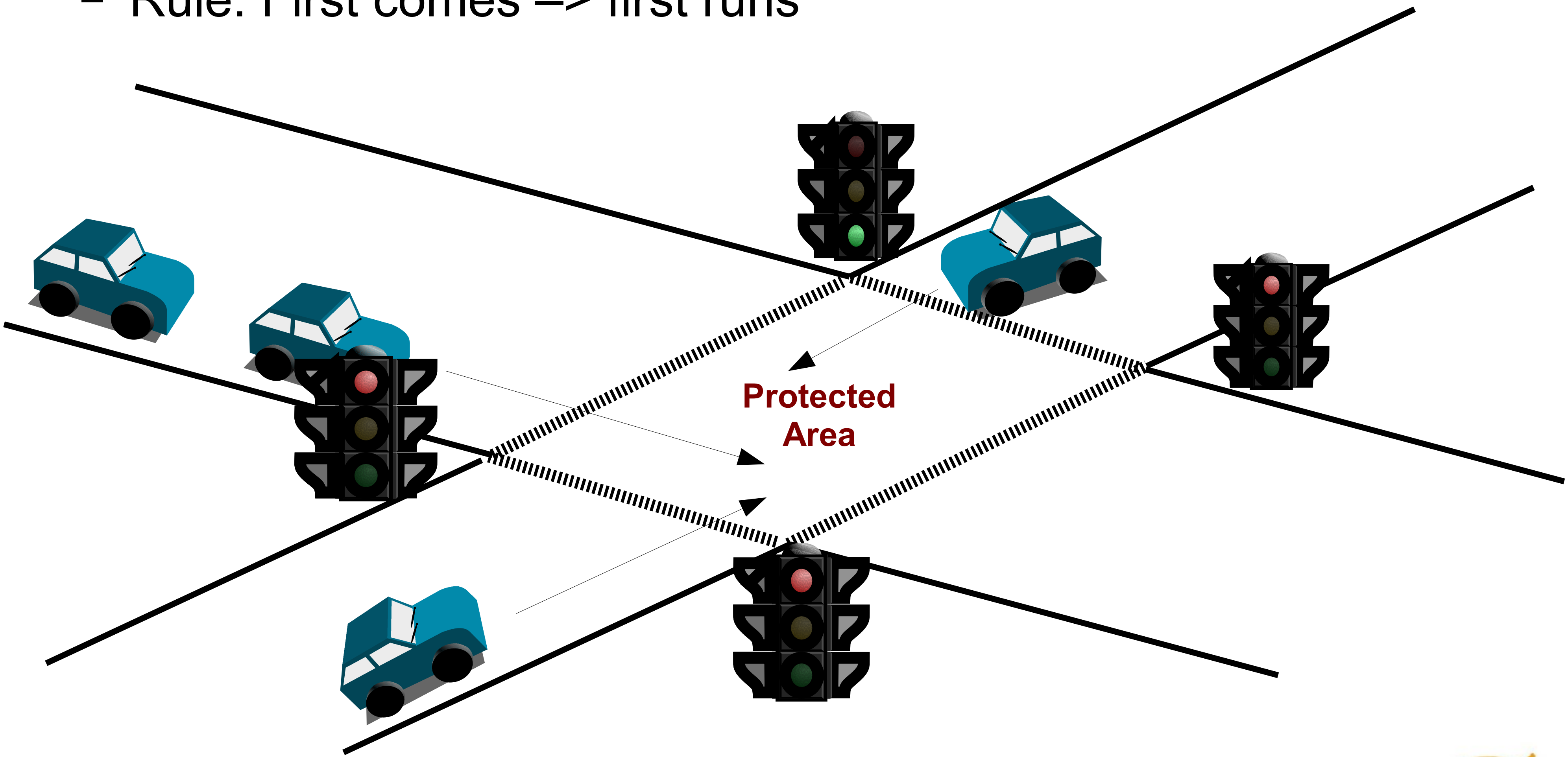
6.3 IPC SEM methods

- *semget* - Get a Set of Semaphores
  - Creates a new set of semaphores returns it's ID
  - Gets the ID of an existing set of semaphoren
- *semop* - Semaphore Operations
  - Performs semaphore operations atomically on a set of semaphores
- *semctl* - Semaphore Control Operations
  - Get and Set semphore values
  - Returns the number of waiting processes
  - Obtains status informations
  - Changes permissions
  - Removes a set of semphores

DEFObonn

6.4 IPC SEM sample no. 1

- Street crossing
  - Rule: First comes –> first runs



6.5 IPC SEM sample no. 1

- Steps

  1. We need one semaphore with one 'traffic light' sem_value
  2. Create or get semaphore by calling semget ()
  3. If new then initialize sem_value with '1' by calling semctl ()
  4. Lock with sem_op equal '-1' by calling semop ()

     if sem_value is '0' then

         process waits until sem_value is set to '1' by another process

     endif

     sem_value becomes '0'

  4. Do your protected work
  5. Unlock with sem_op equal '+1' by calling semop ()

6.5.1 SEM sample program EMEA0002 page 1

EMEA0002.C

```
#define _XOPEN_SOURCE
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <sys/ipc.h>
#include <sys/sem.h>

#define SEM_KEY          0x00004711
#define SEM_SIZE         1
#define SEM_MODE         0666

unsigned int semkey = SEM_KEY;

int main (int argc, *argv []) {

  int            semid, i, rc;
  struct sembuf  semOp;
```


DEFObonn

6.5.2 SEM sample program EMEA0002 page 2

```
system ("%bpx ipcs -a");

/* Get semaphore */
semid = semget (semkey, 0, 0);

/* If not active, create new semphore */
if (semid == -1) {
  semid = semget (semkey, SEM_SIZE,
                  IPC_CREAT | IPC_EXCL | SEM_MODE);
  if (semid == -1) {
    perror ("semget");
    return;
  }
  /* Initialize semaphore */
  rc = semctl (semid, 0, SETVAL, 1);
  if (rc == -1) {
    perror ("semctl");
    return;
  }
}
system ("%bpx ipcs -a");
```

6.5.3 SEM sample program EMEA0002 page 3

```
/* Do locking */
semOp.sem_num = 0;              /* semaphore value number */
semOp.sem_op  = -1;             /* lock                   */
semOp.sem_flg = SEM_UNDO;       /* if aborts, undo change */
rc = semop (semid, &semOp, 1);
if (rc != 0) {
  perror ("semop lock");
  return;
}
system ("%bpx ipcs -w");

/* Do unlocking */
semOp.sem_num = 0;              /* semaphore value number */
semOp.sem_op  = +1;             /* lock                   */
semOp.sem_flg = SEM_UNDO;       /* if aborts, undo change */
rc = semop (semid, &semOp, 1);
if (rc != 0) {
  perror ("semop unlock");
  return;
}
system ("%bpx ipcs -w");
```

6.5.4 SEM sample program EMEA0002 page 4

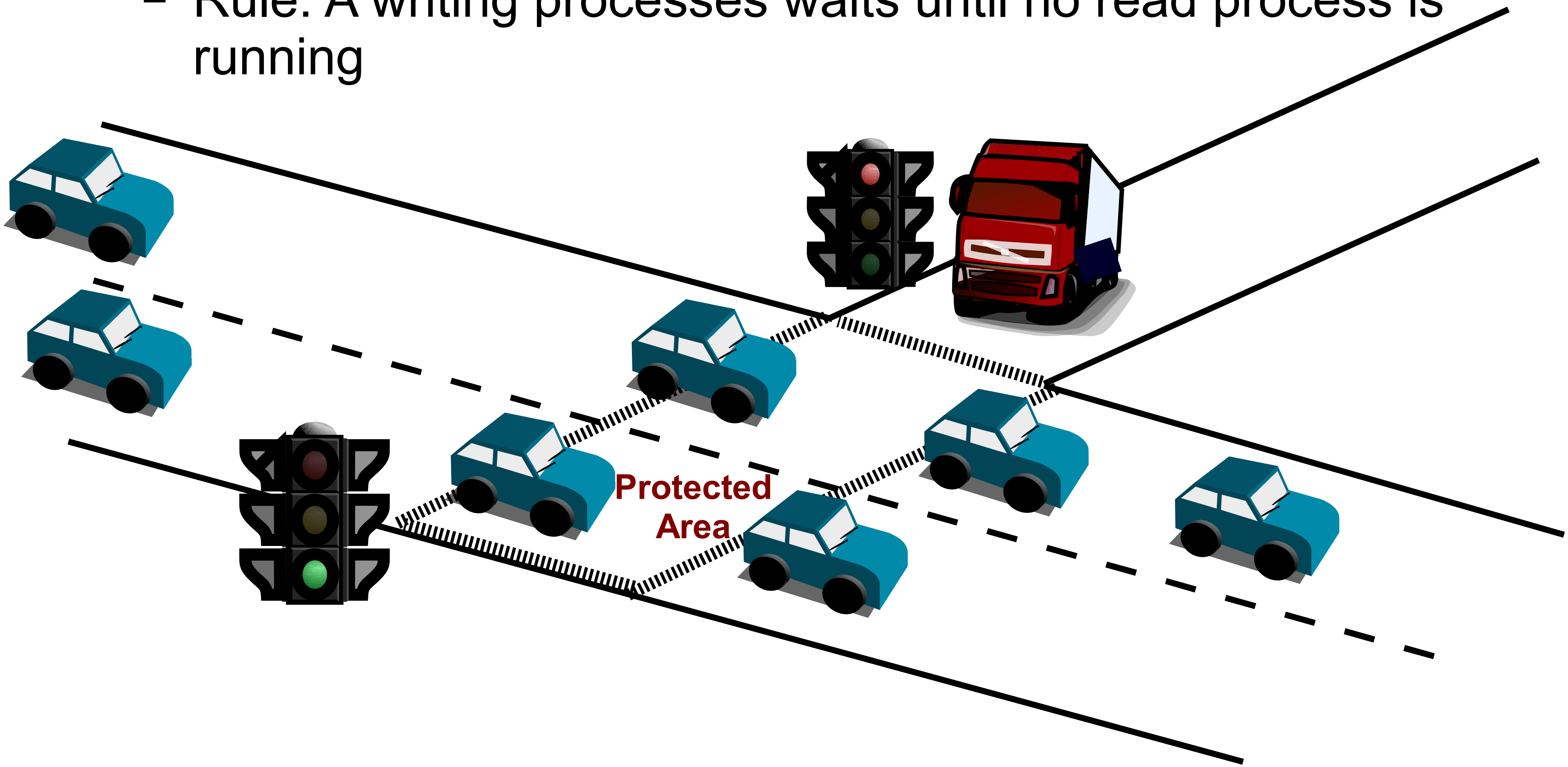
```
   /* Delete permanently the semaphore */
   rc = semctl (semid, 0, IPC_RMID);
   if (rc == -1) {
     perror ("shmctl");
   }
   system ("%bpx ipcs -a");

 } /* main */
```


DEFObonn

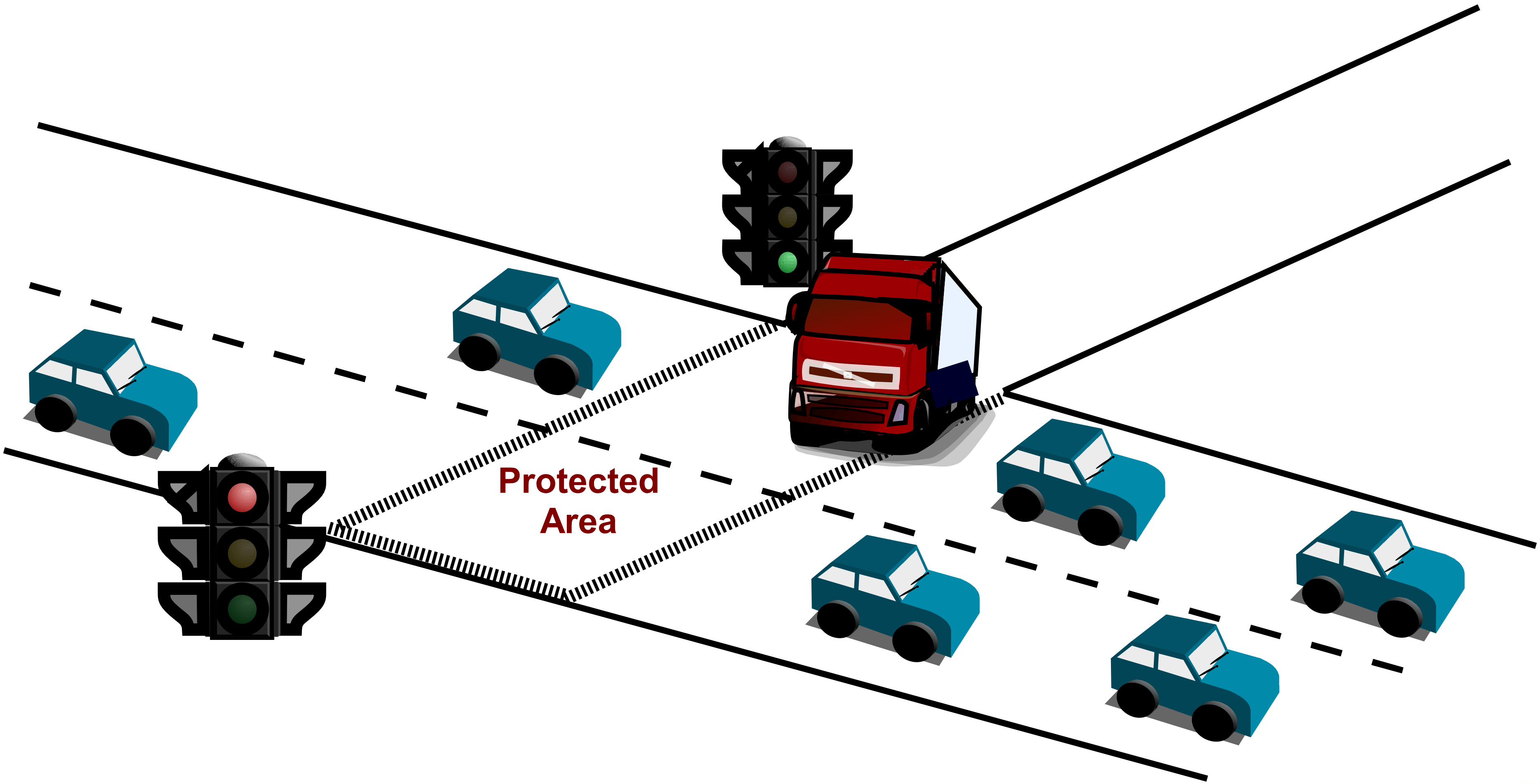
6.6 IPC SEM sample no. 2

- Mainroad (many read processes) and junction (write process)
  - Rule: A writing processes waits until no read process is running



6.7 IPC SEM sample no. 2

- We need a shared global counter for current read processes



6.8 IPC SEM sample no. 2

- Steps

1. We need one semaphore 'W' with one 'traffic light' sem_value
2. We need one semaphore 'R' with two sem_value's
   - The 1st sem_value is used to protect changes of the global counter
   - The 2nd sem_value is used as the counter itself
3. Create or get both semaphores by calling semget ()
4. If new then

   initialize all sem_value's with '1' by calling semctl ()

6.8.1 IPC SEM sample no. 2 - Read Process

- Steps – Read process
  1. Protect and increment global counter

```
long lockRead () {

  struct sembuf semOp [2];
  long          rc;

  semOp [0].sem_num = 0;
  semOp [0].sem_op  = -1; /* lock */
  semOp [0].sem_flg = SEM_UNDO;
  semOp [1].sem_num = 1;
  semOp [1].sem_op  = +1; /* read lock count ++ */
  semOp [1].sem_flg = SEM_UNDO;

  rc = semop (semid_R, &semOp [0], 2); /* lock read now */
  if (rc != 0) {
    printf ("lockRead(1): rc = %d\n", rc);
    return (8);
  }
```

6.8.2 IPC SEM sample no. 2 - Read Process

```
  /* Start of protected resource read counter */
  if (semctl (semidr, 1, GETVAL) == 2) {
    semOp [0].sem_flg = 0;
    rc = semop (semid_W, &semOp [0], 1); /* lock write now */
    if (rc != 0) {
      printf ("lockRead(2): rc = %d\n", rc);
      return (8);
    }
  }
  /* End of protected resource read counter */

  semOp [0].sem_op  = 1; /* unlock */
  semOp [0].sem_flg = SEM_UNDO;
  rc = semop (semid_R, &semOp [0], 1); /* unlock read now */
  if (rc != 0) {
    printf ("lockRead(3): rc = %d\n", rc);
    return (8);
  }
  return (0);
} /* lockRead */
```

6.8.3 IPC SEM sample no. 2 - Read Process

- Steps – Read process
  2. Do your read operation
  3. Protect and decrement global counter

```
long unlockRead () {

  struct sembuf semOp [2];
  long          rc;

  semOp [0].sem_num = 0;
  semOp [0].sem_op  = -1; /* lock */
  semOp [0].sem_flg = SEM_UNDO;
  semOp [1].sem_num = 1;
  semOp [1].sem_op  = -1; /* read lock count -- */
  semOp [1].sem_flg = SEM_UNDO;

  rc = semop (semid_R, &semOp [0], 2); /* lock read now */
  if (rc != 0) {
    printf ("unlockRead(1): rc = %d\n", rc);
    return (8);
  }
```

6.8.4 IPC SEM sample no. 2 - Read Process

```
  /* Start of protected resource read counter */
  semOp [0].sem_op  = 1; /* unlock */

  if (semctl (semidr, 1, GETVAL) == 1) {
    semOp [0].sem_flg = 0;
    rc = semop (semid_W, &semOp [0], 1); /* unlock write */
    if (rc != 0) {
      printf ("unlockRead(2): rc = %d\n", rc);
      return (8);
    }
  }
  /* End of protected resource read counter */

  semOp [0].sem_flg = SEM_UNDO;
  rc = semop (semid_R, &semOp [0], 1); /* unlock read now */
  if (rc != 0) {
    printf ("unlockRead(3): rc = %d\n", rc);
    return (8);
  }
  return (0);
} /* unlockRead */
```


DEFObonn

6.8.5 IPC SEM sample no. 2 – Write Process

- Steps – Write process

  1. Lock semaphore 'W'

```
long lockWrite () {

  struct sembuf semOp;
  long          rc;

  semOp.sem_num = 0;
  semOp.sem_op  = -1; /* lock */
  semOp.sem_flg = SEM_UNDO;

  rc = semop (semid_W, &semOp, 1); /* lock write now */
  if (rc != 0) {
    printf ("lockWrite: rc = %d\n", rc);
    return (8);
  }

  return (0);
} /* lockWrite */
```


DEFObonn

6.8.6 IPC SEM sample no. 2 – Write Process

- Steps – Write process

  2. Do your write operation

  3. Unlock semaphore 'W'

```
long unlockWrite () {

  struct sembuf semOp;
  long          rc;

  semOp.sem_num = 0;
  semOp.sem_op  = 1; /* unlock */
  semOp.sem_flg = SEM_UNDO;

  rc = semop (semid_W, &semOp, 1); /* unlock write now */
  if (rc != 0) {
    printf ("unlockWrite: rc = %d\n", rc);
    return (8);
  }
  return (0);
} /* unlockWrite */
```


DEFObonn

7. Your Own IPC Utility

- __*getipc* - Query Interprocess Communications
  - The __getipc() function provides means for obtaining information about the status of interprocess communications (IPC) resources, message queues, semaphores and shared memory.

<u>EMEA0003.C</u>

```
#define _XOPEN_SOURCE
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <errno.h>
#include <sys/ipc.h>
#include <sys/__getipc.h>
#include <sys/shm.h>
#include <sys/sem.h>
```


DEFObonn

7.1 Your Own IPC Utility

```
int main (int argc, *argv []) {

  long              token = 0;
  long              rc = 0;
  IPCQPROC          buffer;
  int               semval;

  printf ("... analyzing IPCS records ...\n");
  do {
    rc = __getipc (token, &buffer, sizeof (buffer), IPCQALL);
    token = rc;
    if (token == 0) {
      break;
    }
```


DEFObonn

7.2 Your Own IPC Utility

```
    if (memcmp (buffer.shm.ipcqtype, "ISEM", 4) == 0) {
      printf ("\nSEMAPHORE: Key = %8.8p, ID = %d\n\n",
              buffer.sem.ipcqkey, buffer.sem.ipcqmid);

      /* semval */
      semval = semctl (buffer.sem.ipcqmid, 0, GETVAL);
      printf ("Actual value (0) = %d\n", semval);
      semval = semctl (buffer.sem.ipcqmid, 1, GETVAL);
      printf ("Actual value (1) = %d\n\n", semval);
    }

    else if (memcmp (buffer.shm.ipcqtype, "ISHM", 4) == 0) {
      printf ("\nSHARED MEMORY: Key=%8.8p, Id=%d\n\n",
              buffer.shm.ipcqkey, buffer.shm.ipcqmid);
      printf ("  USE count (attach - detach) = %10d\n",
              buffer.shm.ipcqacnt);
    }

  } while (rc != 0);

} /* main */
```


DEFObonn

8. IPC Documentation

Documentation

8.1 IPC and XL C/C++

- Documentation
  - z/OS XL C/C++ Run-Time Library Reference
    - Dokument Number SA22-7821-09
    - Program Number 5694-A01
  - http://publibfp.boulder.ibm.com/cgi-bin/bookmgr/BOOKS/edclb180/CCONTENTS
  - See
    - Chapter 2,
      sys/ipc.h, sys/sem.h, sys/shm.h and sys/__getipc.h
    - Chapter 3,
      shm*, sem* and __getipc

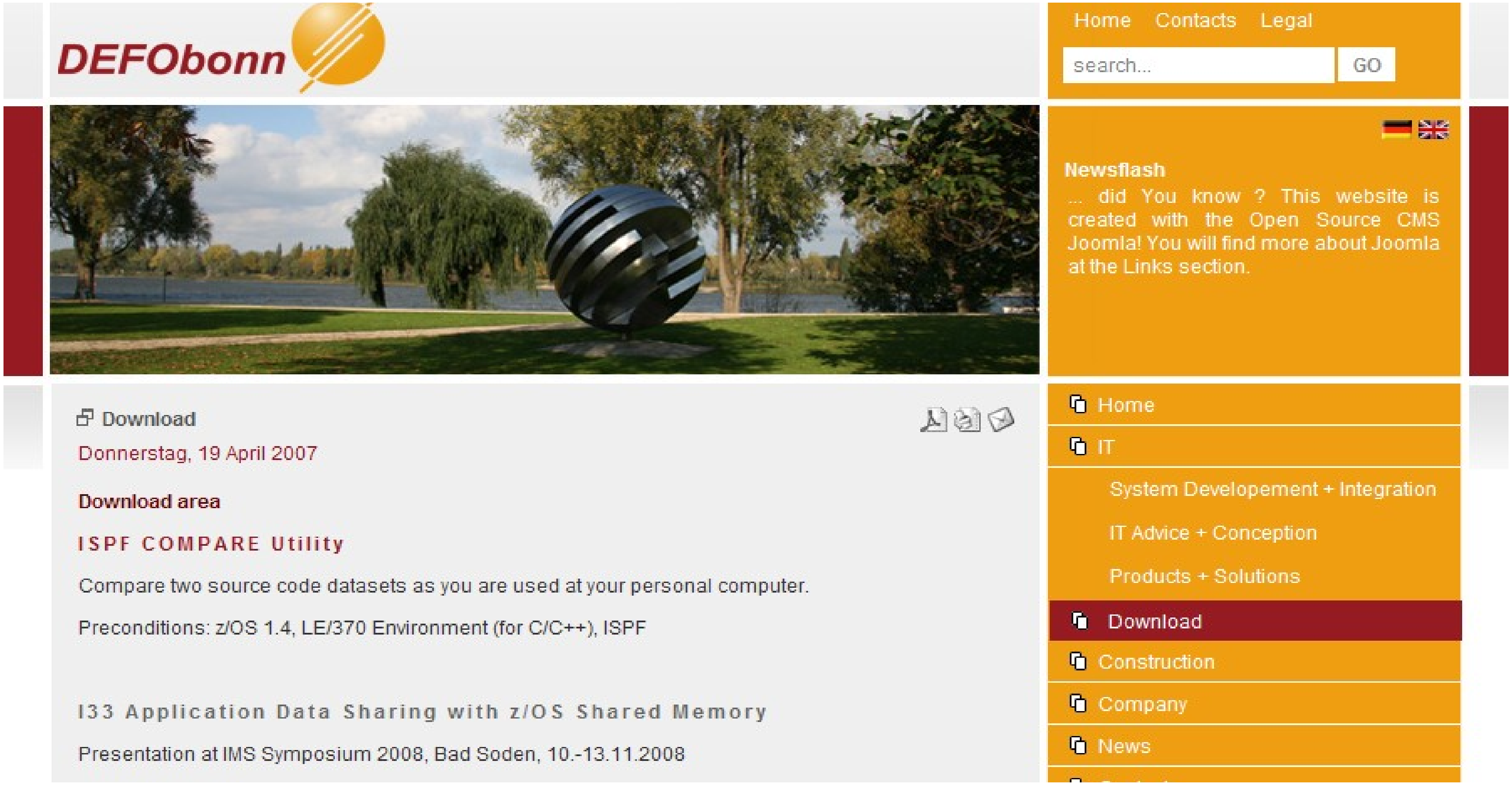
DEFObonn

8.2 IPC and Assembler Unix Services

- Documentation
  - z/OS UNIX System Services
    Programming: Assembler Callable Services reference
    - Dokument Number SA22-7803-10
    - Program Number 5694-A01
  - http://publibfp.boulder.ibm.com/cgi-bin/bookmgr/BOOKS/bpxzb180/CCONTENTS
  - See
    - Chapter 2,
      sem*, shm* and w_getipc
    - Appendix B,
      BPXYIPCP, BPXYMODE, BPXYSEM and BPXYIPCQ

9. Presentation Download

You can download this presentation here:
http://www.defobonn.com/content/view/41/61/lang,en/



DEFObonn

10. Your Contact

Detlef Dewitz

DEFObonn GmbH
Argelanderstr. 183
53115 Bonn
Germany

Phone	+49 (0)228 9146013
Fax	+49 (0)228 215180
Web	http://www.defobonn.com
email	D.Dewitz@defobonn.de

DEFObonn

11. Discussion

Questions ?
Comments ?

DEFObonn