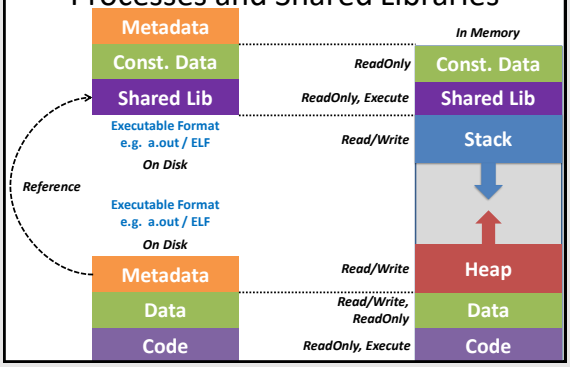
Paging and Shared Memory

Dr Andrew Scott

a.scott@lancaster.ac.uk

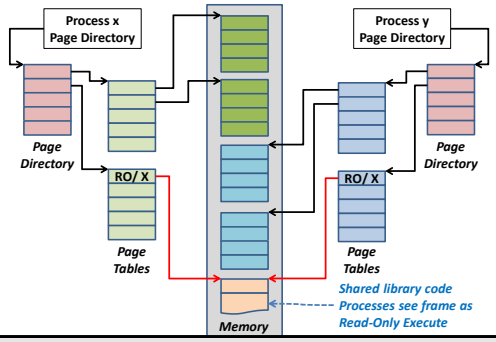
1

Processes and Shared Libraries



2

Shared Libraries (/memory)



3

Dynamic Libraries

```
#include <dlfcn.h>
#include <stdio.h>

main( )
{
  void  *  libc;
  double (*func)(double); // matches 'signature' of sqrt( )
  double   res;

  if( libc = dlopen("/lib/x86_64-linux-gnu/libm.so.6",RTLD_LAZY) )
  {
    func = dlsym( libc,"sqrt" ); // Find function entry point

    res = (*func)(144);          // Call function - sqrt(144)

    printf ( "Sqrt returned %lf\n", res );
  }
  else { printf ( "Couldn't open library\n" ); }

}
```

RTLD_LAZY: Perform *lazy binding*. Only resolve symbols as the code that references them is executed. If the symbol is never referenced, then it is never resolved.

4

Creating a Shared Memory Area

```
#define LENGTH      sizeof(char) /* Size of shared memory area  */
#define SHM_FILE    "/mypage"    /* Name for area in filesystem */

int
main ( ) {
   int     od;
   int     i;
   int     child_id;
   char  * f;

   od = shm_open(SHM_FILE, O_CREAT | O_RDWR, S_IRUSR | S_IWUSR);
   ftruncate ( od, LENGTH );

   f = mmap(NULL, LENGTH, PROT_READ|PROT_WRITE, MAP_SHARED, od, 0);

   ...  // Use shared memory area

   munmap ( f, LENGTH );     // tidy up
   shm_unlink ( SHM_FILE );
}
```

How /mypage should be created/ opened

File access permissions for new file /mypage

5

Example: *using shared char * f*

Process 1

```
...
for ( i = 'a'; i <='z'; i++ ) {
      *f = i;
      usleep ( 1000 );
   }
   *f = 0;  // End of string
}
```

Process 2

```
...
do {
   i = *f;  putchar ( i );

   // Wait for change...
   while ( i == *f ) usleep( 500 );

} while ( *f != 0 );

putchar ( '\n' );
```

Both programs use previous set-up code using same name for shared memory region.

Programs pass data by referencing shared variables/ objects.

6

Memory Mapping a File

```
#define LENGTH  4096

int
main ( ) {
   int     fd;
   int     i;
   char  * f;

   if ( (fd = open ("testfile.txt", O_RDWR, 0)) < 0 ) exit (1);

   f = mmap( NULL, LENGTH, PROT_READ | PROT_WRITE,
                           MAP_FILE | MAP_SHARED, fd, 0 );

   for (i = 0; i < 16; i++)
      f[i + 16] = f[i];    // Changes to memory reflected in file

   munmap (f, LENGTH);
   close (fd);
}
```

7