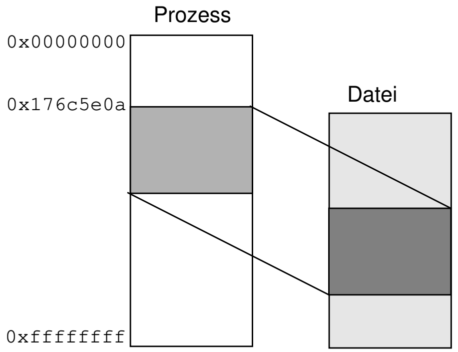
50 Überblick über die 11. Übung

- Besprechung 7. Aufgabe (port_forward)
- Memory Mapped Files (mmap)
- Beispiel: Revision Control System - RCS

51 Memory Mapped Files

- Mit mmap kann eine Datei in den Adressraum eines Prozesses eingeblendet werden.



51.1 mmap (Speicherbereich)

```
#include <unistd.h>
#include <sys/mman.h>

void * mmap(void *start, size_t length, int prot,
            int flags, int fd, off_t offset);
```

- Der Parameter **`start`** gibt eine "Wunschadresse" an, ab welcher Adresse die Daten im Adressraum erscheinen sollen. Die Adresse muss ein vielfaches der Speicherseitengrösse sein. Anstelle einer gültigen Adresse kann auch 0 angegeben werden.
- Als Rückgabewert wird die tatsächlich Adresse zurückgeliefert, oder MAP_FAILED (-1) im Fehlerfall.
- Der Parameter **`length`** gibt die Länge des Speicherbereiches an.

51.2 mmap (Zugriffsrechte)

```
#include <unistd.h>
#include <sys/mman.h>

void * mmap(void *start, size_t length, int prot,
            int flags, int fd, off_t offset);
```

- Der Parameter **`prot`** gibt die Zugriffsrechte auf den Speicher an. Die Zugriffsrechte auf die Datei müssen passen.
- Der Wert setzt sich aus einer Bitweisen-oder-Verknüpfung folgender Makros zusammen.
  - ◆ **PROT_READ** Der Speicherinhalt darf gelesen werden.
  - ◆ **PROT_WRITE** Der Speicherinhalt darf verändert werden.
  - ◆ **PROT_EXEC** Der Speicherinhalt darf ausgeführt werden.

51.3 mmap (Flags)

```
#include <unistd.h>
#include <sys/mman.h>

void * mmap(void *start, size_t length, int prot,
            int flags, int fd, off_t offset);
```

- Mit dem Parameter **`flags`** kann das Verhalten von mmap beeinflusst werden
- Im POSIX-Standard sind folgende Makros definiert.
  - **MAP_FIXED** Wenn die mit **`start`** festgelegte Adresse nicht vergeben werden kann, wird ein Fehler zurückgeliefert.
  - **MAP_SHARED** Der pysikalische Speicher wird von verschiedenen Prozessen gemeinsam genutzt. Änderungen werden in allen Prozessen sofort sichtbar.
  - **MAP_PRIVATE** Bei Änderungen bekommt der jeweilige Prozess seine eigene Kopie des Speichers.

51.4 mmap (Datei)

```
#include <unistd.h>
#include <sys/mman.h>

void * mmap(void *start, size_t length, int prot,
            int flags, int fd, off_t offset);
```

- Der Parameter **`fd`** ist ein gültiger Filedescriptor auf die Datei.
- Mit **`offset`** kann die Startposition innerhalb der Datei angegeben werden.

51.5 munmap

```
#include <unistd.h>
#include <sys/mman.h>

int munmap(void *start, size_t length);
```

- **`munmap`** macht das einblenden der Datei in den entsprechenden Speicherbereicht wieder rückgängig.
- Der Speicherinhalt wird in die Datei geschrieben.
- Das Schließen des Filedescriptors hat keine Auswirkung.
- Beim Beenden eines Prozesses werden alle eingeblendeten Speicherbereiche wird freigegeben.
- Im Fehlerfall wird -1 zurückgeliefert, ansonsten 0.

51.6 msync

```
#include <unistd.h>
#include <sys/mman.h>

int msync(const void *start, size_t length, int flags);
```

- **`msync`** schreibt den Inhalt eines eingeblendeten Speicherbereichs in die Datei.
- Die **`flags`** beeinflussen das Verhalten von **`msync`** wie folgt:
  - **MS_ASYNC** Der Aufruf kehrt sofort zurück.
  - **MS_SYNC** Der Aufruf blockiert bis die Daten auf die Platte geschrieben sind.
  - **MS_INVALIDATE** Markiert auch andere eingeblendete Seiten dieser Datei als Invalidate, sodass diese den neuen Inhalt bekommen.
- Im Fehlerfall wird -1 zurückgeliefert, ansonsten 0.

51.7 Beispiel lesen

```
typedef struct {
    char name[50];
    char str[50];
    char plz_ort[50];
} adr_t;

void lesen() {
  int  fd;
  adr_t *adr;

  if ((fd=open("datei", O_RDONLY))<0)) { ... }

  adr=(adr_t*)mmap(0,sizeof(adr_t),
                   PROT_READ,MAP_SHARED,fd,0);
  if (adr==-1) { ... }
  close(fd);

  printf("%s/n%s\n\n%s\n",adr->name,adr->str,adr->plz_ort);
  munmap(adr,sizeof(adr_t));
}
```


51.8 Beispiel schreiben

```
#define FILE_MODE ...
void schreiben() {
  int  fd;
  adr_t *adr;

  fd=open("datei",O_RDWR|O_CREAT|O_EXECL,FILE_MODE);
  if (fd<0) { ... }
  lseek(fd, sizeof(adr_t)-1, SEEK_SET);
  write(fd, "", 1);

  adr=(adr_t*)mmap(0,sizeof(adr_t),
         PROT_READ|PROT_WRITE,MAP_SHARED,fd,0);
  if (adr==-1) { ... }
  close(fd);

  strcpy(adr->name,"...");
  strcpy(adr->strasse,"...");
  strcpy(adr->plz_ort,"...");
  munmap(adr,sizeof(adr_t));
}
```


52 Beispiel: Revision Control System

52.1 Wiederholung (Übung 4)

- RCS ist ein Versionskontrollsystem, das
  - Änderungen an Dateien mit dem Namen des Ändernden, dem Zeitpunkt und einem Kommentar speichert
  - Zugriffe auf Versionen kontrolliert und koordiniert
  - eindeutige Identifizierung verwendeter Versionen erlaubt
  - redundante Speicherung von Versionen vermeidet

52.2 Verwalten einer Datei mit RCS

- Es ist sinnvoll im Arbeitsverzeichnis ein Unterverzeichnis mit dem Namen "**RCS**" anzulegen, da dort dann die von RCS erzeugten Datein abgelegt werden.
- Anschließend die Datei mit `ci -u <dateiname>` einchecken.
- Aufrufsyntax (nur die wichtigsten Optionen angegeben!):

  `ci [-rrev] [-lrev] [-urev] filename ...`

  -r*rev* die neue Version erhält Version ***rev***
  - ***rev*** muß größer als die letzte existierende Version sein
  - soll eine neue *Release* erzeugt werden, genügt die Angabe der *Release*-Nummer (z. B. **-r5**)

  -l*rev* wie **ci -r**, anschließend wird automatisch ein **co -l** durchgeführt

  -u*rev* wie **ci -r**, anschließend erfolgt ein **co**

52.2 Verwalten einer Datei mit RCS

- Zum Bearbeiten kann die Datei mit **`co -l <dateiname>`** wieder ausgecheckt werden.
- Aufrufsyntax (nur die wichtigsten Optionen angegeben!):

  `co [-rrev] [-lrev] [-urev] filename ...`

  `-r`*rev* extrahiert die neueste Version, der Versionsnummer kleiner oder gleich ***rev*** ist

  `-l`*rev* wie **`co -r`**, die extrahiert Version wird durch den Aufrufer gesperrt

  `-u`*rev* wie **`co -r`**, falls eine Sperre der Version durch den Aufrufer existiert, wird diese aufgehoben

52.3 Beispiel: Hello World

```
#include <stdio.h>

static char rcsid[] = "$Id$";
static char rcsrev[] = "$Revision$";

int main(int argc,char* argv[]) {
        printf("Hello World (%s)\n",rcsrev);
        return 0;
}
```