Πανεπιστήμιο Δυτικής Μακεδονίας
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Λειτουργικά Συστήματα

Ενότητα 10: Επικοινωνία διεργασιών με ουρές μηνυμάτων. Επικοινωνία διεργασιών με κοινή μνήμη. Επικοινωνία διεργασιών με απεικόνιση αρχείου στη μνήμη. Ο προσομοιωτής διεργασιών SOSSIM.

Επιβλέπων: Δασυγένης Μηνάς
Παυλίδου Ελένη

Δρ. Μηνάς Δασυγένης
mdasyg@ieee.org
Εργαστήριο Λειτουργικών Συστημάτων
http://arch.ece.uowm.gr/courses/os/

Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ψηφιακά Μαθήματα στο Πανεπιστήμιο Δυτικής Μακεδονίας**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Η συνάρτηση msgget() (1/2)

- Η συνάρτηση `msgget()` αρχικοποιεί μια νέα ουρά μηνυμάτων:

```
int msgget(key_t key, int msgflg)
```

- Επίσης επιστρέφει τον αριθμό της ουράς (msqid) που αντιστοιχεί στο όρισμα `key`. Η τιμή του ορίσματος `msgflg` πρέπει να είναι ένας οκταδικός αριθμός που ορίζει τις άδειες πρόσβασης και ελέγχου της ουράς.

- Ο κώδικας που ακολουθεί επιδεικνύει τη χρήση της συνάρτησης `msgget()`.

Η συνάρτηση msgget() (2/2)

```
#include <sys/ipc.h>;
#include <sys/msg.h>;
key_t key;  /* key to be passed to msgget() */
int msgflg  /* msgflg to be passed to msgget() */
int msqid;  /* return value from msgget() */
key = …
msgflg = …
if ((msqid = msgget(key, msgflg)) == -1)
{
perror("msgget: msgget failed");
exit(1);
} else
(void) fprintf(stderr, &ldquo;msgget
succeeded");
```

Οι συναρτήσεις msgsnd() και msgrcv() (1/4)

- Οι συναρτήσεις `msgsnd()` και `msgrcv()` στέλνουν και λαμβάνουν μηνύματα, αντίστοιχα:

```
int msgsnd(int msqid, const void *msgp,
size_t msgsz, int msgflg);


int msgrcv(int msqid, void *msgp, size_t
msgsz, long int msgtype,int msgflg);
```

Οι συναρτήσεις msgsnd() και msgrcv() (2/4)

- Το όρισμα `msqid` πρέπει να είναι το ID μιας υπάρχουσας ουράς. Το όρισμα `msgp` είναι δείκτης σε μια δομή που περιέχει το μήνυμα, όπως στο παράδειγμα που ακολουθεί:

```
struct mymsg {
long int msgtype; /* message type */
char mtext[1]; /* message text of length 1 */
}
```

- Το όρισμα `msgsz` ορίζει το μέγεθος του μηνύματος σε bytes.
- Το όρισμα `msgtype` του παραλήπτη είναι ίδιο με το μέλος `msgtype` της δομής `mymsg` του αποστολέα.

Οι συναρτήσεις msgsnd() και msgrcv() (3/4)

- Το όρισμα `msgflg` ορίζει τις λειτουργίες που πρέπει να κάνει η συνάρτηση αν ισχύει μια από τις παρακάτω συνθήκες:

- Ο αριθμός των bytes στην ουρά ξεπερνά ένα όριο που θέτει το σύστημα.

- Ο συνολικός αριθμός των μηνυμάτων σε όλες τις ουρές του συστήματος είναι ίσος με ένα όριο που θέτει το σύστημα.

Οι συναρτήσεις msgsnd() και msgrcv() (4/4)

- Οι λειτουργίες είναι οι παρακάτω:

• Αν το (`msgflg & IPC_NOWAIT`) είναι 0, η καλούσα διεργασία θα ανασταλεί μέχρι να συμβεί κάτι από τα παρακάτω:

✓ Η συνθήκη που προκάλεσε την αναστολή δεν ισχύει πλέον, οπότε το μήνυμα στέλνεται.
✓ Η ουρά με ID `msqid` διαγράφεται από το σύστημα, οπότε το `errno` τίθεται ίσο με EIDRM και η συνάρτηση επιστρέφει -1.
✓ Η καλούσα διεργασία λαμβάνει ένα σήμα, οπότε το μήνυμα δεν στέλνεται αλλά η διεργασία συνεχίζει για να χειριστεί το σήμα.
✓ Αν το (`msgflg & IPC_NOWAIT`) είναι μη-μηδενικό, το μήνυμα δεν θα σταλεί και η καλούσα διεργασία θα επιστρέψει άμεσα.

Η συνάρτηση msgctl() (1/2)

- Η συνάρτηση `msgctl()` επιτρέπει στον ιδιοκτήτη ή το δημιουργό μιας ουράς μηνυμάτων (ή σε όποια διεργασία έχει τα κατάλληλα δικαιώματα) να τροποποιεί τα δικαιώματα πρόσβασης και άλλα χαρακτηριστικά της.

- Η συνάρτηση `msgctl()` ορίζεται ως εξής:

```
int msgctl(int msqid, int cmd, struct
msqid_ds *buf)
```

- Το όρισμα `msqid` πρέπει να είναι το ID μιας υπάρχουσας ουράς.

Η συνάρτηση msgctl() (2/2)

➢ Το όρισμα `cmd` είναι ένα από τα:

- **IPC_STAT**-- Επιστρέφει πληροφορίες για την κατάσταση της ουράς στη δομή που δείχνει ο buf.
  Η καλούσα διεργασία πρέπει να έχει δικαίωμα ανάγνωσης.
- **IPC_SET**-- Ρυθμίζει το ID του ιδιοκτήτη και της ομάδας, τα δικαιώματα και το μέγεθος της ουρά (σε bytes). Η καλούσα διεργασία πρέπει να έχει δικαίωμα ιδιοκτήτη ή διαχειριστή.
- **IPC_RMID**-- Διαγραφή της ουράς που ορίζεται από το όρισμα `msqid`. Η καλούσα διεργασία πρέπει να έχει δικαίωμα ιδιοκτήτη ή διαχειριστή.

Η συνάρτηση shmget() (1/4)

- Η συνάρτηση `shmget()` χρησιμοποιείται για τη δημιουργία ή πρόσβαση σε ένα τμήμα μοιραζόμενης μνήμης. Ορίζεται ως:

```
int shmget(key_t key, size_t size, int shmflg);
```

- Το όρισμα `key` είναι συνήθως ένας αριθμός που συσχετίζεται με το ID του τμήματος μοιραζόμενης μνήμης. Το όρισμα `size` καθορίζει το μέγεθος του δημιουργούμενου τμήματος σε bytes. Το όρισμα `shmflg` καθορίζει τις αρχικές άδειες πρόσβασης και ιδιοκτησίας.

Η συνάρτηση shmget() (2/4)

- Αν η κλήση της συνάρτησης επιτύχει, επιστρέφει το ID του τμήματος. Η ίδια συνάρτηση χρησιμοποιείται από άλλες διεργασίες για να λάβουν το ID ενός υπάρχοντος τμήματος.

- Το απόσπασμα που ακολουθεί δείχνει τη χρήση της συνάρτησης `shmget()`:

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
…
```

Συνέχεια στην επόμενη διαφάνεια.

Η συνάρτηση shmget() (3/4)

```
key_t key;  /* key to be passed to shmget() */
int shmflg;  /* shmflg to be passed to shmget() */
int shmid;  /* return value from shmget() */
int size;  /* size to be passed to shmget() */
…
key = …
size = …
shmflg) =…
```

Συνέχεια στην επόμενη σελίδα.

Η συνάρτηση shmget() (3/3)

```
if ((shmid = shmget (key, size, shmflg)) == -
1)
{
perror("shmget: shmget failed");
exit(1);
 }
else
{
(void) fprintf(stderr, "shmget: shmget
returned %d\n", shmid);
exit(0);
}
```

Η συνάρτηση shmat() (1/2)

- Αφού δημιουργηθεί, το τμήμα μοιραζόμενης μνήμης προσαρτάται στο χώρο διευθύνσεων μιας διεργασίας με τη συνάρτηση `shmat()`. Μπορεί να αποπροσαρτηθεί με τις συναρτήσεις `shmdt()` και `shmop()`.

- Οι συναρτήσεις `shmat()` και `shmdt()` χρησιμοποιούνται για τη προσάρτηση και αποπροσάρτηση τμημάτων μοιραζόμενης μνήμης. Ορίζονται ως ακολούθως:

```
 void *shmat(int shmid, const void *shmaddr,
int shmflg);
```

Η συνάρτηση shmat() (2/2)

```
int shmdt(const void *shmaddr);
```

- Η συνάρτηση `shmat()` επιστρέφει ένα δείκτη τον `shmaddr`, στην αρχή του τμήματος μοιραζόμενης μνήμης που σχετίζεται με ένα έγκυρο `shmid`.
  Η συνάρτηση `shmdt()` αποπροσαρτά το τμήμα μοιραζόμενης μνήμης που δείχνει ο δείκτης `shmaddr`.

Η συνάρτηση shmdt() (1/2)

- Αφού χρησιμοποιήσουμε την κοινή μνήμη, το επόμενο βήμα είναι να αποσπάσουμε τη μνήμη από τη διεργασία. Αυτό επιτυγχάνεται με την κλήση της `shmdt()`.

- Οι συναρτήσεις `shmat()` και `shmdt()` χρησιμοποιούνται για τη προσάρτηση και αποπροσάρτηση τμημάτων μοιραζόμενης μνήμης.

Η συνάρτηση shmdt() (2/2)

Ορίζονται ως ακολούθως:

```
void *shmat(int shmid, const void *shmaddr,
int shmflg);

int shmdt(const void *shmaddr);
```

- Η συνάρτηση `shmat()` επιστρέφει ένα δείκτη, τον `shmaddr`, στην αρχή του τμήματος μοιραζόμενης μνήμης που σχετίζεται με ένα νόμιμο `shmid`. Η συνάρτηση `shmdt()` αποπροσαρτά το τμήμα μοιραζόμενης μνήμης που δείχνει ο δείκτης `shmaddr`.

Η συνάρτηση shmctl() (1/4)

- Ο δημιουργός και ιδιοκτήτης του τμήματος μοιραζόμενης μνήμης μπορεί να παραχωρήσει δικαιώματα σε άλλους με τη συνάρτηση `shmctl()`. Τα δικαιώματα αυτά μπορούν να ανακληθούν. Άλλες διεργασίες, με τα κατάλληλα δικαιώματα, μπορούν να εκτελέσουν διάφορες λειτουργίες ελέγχου με την ίδια συνάρτηση `shmctl()`.

Η συνάρτηση shmctl() (2/4)

- Ορίζεται ως εξής:

```
int shmctl(int shmid, int cmd, struct
shmid_ds *buf);
```

- Τα δικαιώματα ορίζονται με βάση το `shmid` του τμήματος. Το όρισμα `cmd` έχει μια από τις παρακάτω τιμές:

- **SHM_LOCK**-- Κλείδωμα του τμήματος μοιραζόμενης μνήμης. Η καλούσα διεργασία πρέπει να έχει δικαίωμα ιδιοκτήτη, δημιουργού ή διαχειριστή.

Η συνάρτηση shmctl() (3/4)

- **SHM_UNLOCK**-- Ξεκλείδωμα του τμήματος μοιραζόμενης μνήμης. Η καλούσα διεργασία πρέπει να έχει δικαίωμα ιδιοκτήτη, δημιουργού ή διαχειριστή.

- **IPC_STAT**-- Επιστροφή των πληροφοριών κατάστασης που περιέχονται στη δομή ελέγχου και τοποθετούνται στον απομονωτή που δείχνει ο *buf.* Η καλούσα διεργασία πρέπει να έχει δικαίωμα ανάγνωσης.

- **IPC_SET**-- Καθορισμός δικαιωμάτων πρόσβασης. Η καλούσα διεργασία πρέπει να έχει δικαίωμα ιδιοκτήτη, δημιουργού ή διαχειριστή.

Η συνάρτηση shmctl() (4/4)

- **IPC_RMID**-- Διαγραφή του τμήματος μοιραζόμενης μνήμης. Η καλούσα διεργασία πρέπει να έχει δικαίωμα ιδιοκτήτη, δημιουργού ή διαχειριστή.

❖ Η δομή που δείχνει το buf είναι του τύπου struct shmid_ds που ορίζεται στη βιβλιοθήκη `sys/shm.h`.

Δημιουργία και χρήση απεικονίσεων (1/5)

- Η συνάρτηση `mmap()` ορίζει μια απεικόνιση ενός ονοματισμένου αρχείου στο χώρο διευθύνσεων. Είναι πολύ απλή και βασική στην απεικόνιση μνήμης.

- Πρώτα ανοίγουμε το αρχείο με συνάρτηση `open()`.
- Μετά απεικονίζουμε με την συνάρτηση `mmap()` με κατάλληλες επιλογές πρόσβασης και διαμοιρασμού.

Δημιουργία και χρήση απεικονίσεων (2/5)

- Η συνάρτηση `mmap()` ορίζεται ως ακολούθως:

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

caddr_t mmap(caddr_t addr, size_t len, int
prot, int flags, int fildes, off_t off);
```

Δημιουργία και χρήση απεικονίσεων (3/5)

- Η απεικόνιση που ορίζεται από την `mmap()` αντικαθιστά παλιότερες απεικονίσεις για το συγκεκριμένο χώρο διεύθυνσης. Οι σημαίες `flags` MAP_SHARED και MAP_PRIVATE καθορίζουν τον τύπο απεικόνισης, μπορεί δε να οριστεί μόνο μία σημαία. Η MAP_SHARED καθορίζει ότι το απεικονιζόμενο αντικείμενο μπορεί να τροποποιηθεί με εγγραφή. Η MAP_PRIVATE καθορίζει ότι μια αρχική εγγραφή στο απεικονιζόμενο αντικείμενο δημιουργεί ένα αντίγραφο της σελίδας και όλες οι εγγραφές αναφέρονται σε αυτή τη σελίδα. Στο τέλος αντιγράφονται μόνο σελίδες που έχουν τροποποιηθεί.

Δημιουργία και χρήση απεικονίσεων (4/5)

- Στην περίπτωση `fork()`, ο τύπος απεικόνισης (το `flag`) διατηρείται. Ο περιγραφέας αρχείου μιας συνάρτησης `mmap()` δεν χρειάζεται να παραμένει ανοικτός μετά την απεικόνιση. Αν ο περιγραφέας κλείσει η απεικόνιση παραμένει μέχρι να αναιρεθεί με τη συνάρτηση `munmap()` ή με ένα νέο `mmap()` που αντικαθιστά.

- Ο κώδικας που ακολουθεί επιδεικνύει μια χρήση των παραπάνω συναρτήσεων.

Δημιουργία και χρήση απεικονίσεων (5/5)

- Δημιουργεί μια περιοχή αποθήκευσης για ένα πρόγραμμα, σε μια διεύθυνση που επιλέγει το σύστημα:

```
int fd;
caddr_t result;
if ((fd = open("/dev/zero", O_RDWR)) == -1)
return ((caddr_t)-1);

result = mmap(0, len, PROT_READ|PROT_WRITE,
MAP_SHARED, fd, 0);
(void) close(fd);
```

Ο προσομοιωτής διεργασιών SOSIM (1/2)

- Ο προσομοιωτής SOsim υλοποιεί τις διεργασίες ως Process Cntrl Blcks (PCBs), όπου διατηρούνται οι πληροφορίες της κάθε διεργασίας. Ο χρονοπρογραμματισμός είναι η διαδικασία της απόφασης επιλογής της διεργασίας που θα «κερδίσει» τη CPU. Υλοποιείται με: Προεκτοπιστικούς αλγορίθμους (preemptive algrithms).

- Ο προσομοιωτής SOSIM έχει 6 παράθυρα: Παράθυρο SOsim Cnsle, παράθυρο Process Manager, παράθυρο Lg, παράθυρο Statistics παράθυρο Select a Process, παράθυρο Memry Manager.

Ο προσομοιωτής διεργασιών SOSIM (2/2)

- Η λειτουργία του παραθύρου SOSim Cnsle: Δημιουργία νέας διεργασίας (Prcess / Create). Ορισμός κατηγορίας διεργασίας (CPU-bund, I/O bund κ.τ.λ) I/O-1, I/O-2, I/O-3 : αν έχουμε ορίσει χρονοπρογραμματισμό με δυναμική προτεραιότητα (dynamic pririty scheduling), τα -1, -2, -3 δηλώνουν την αύξηση της προτεραιότητας της διεργασίας κατά +1, +2, +3 MIX-1: MIX-2: η διεργασία καταναλώνει το χρόνο της τόσο σε I/O όσο και στη CPU ακολουθία καταστάσεων διεργασίας: ready->running->ready by time->i/o pririty bst +1 (όταν έχει οριστεί χρονοπρογραμματισμός με δυναμική προτεραιότητα).

Τέλος Ενότητας

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης