Software Metrics

Testing MINIX 3

ΛΕΙΤΟΥΡΓΙΚΑ ΣΥΣΤΗΜΑΤΑ

ΔΕΥΤΕΡΗ ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ

- **Ονόματα μελών**

1. Γρηγοριάδης Νικόλαος AM:3208
2. Κολιάτος Δημήτριος AM:3252

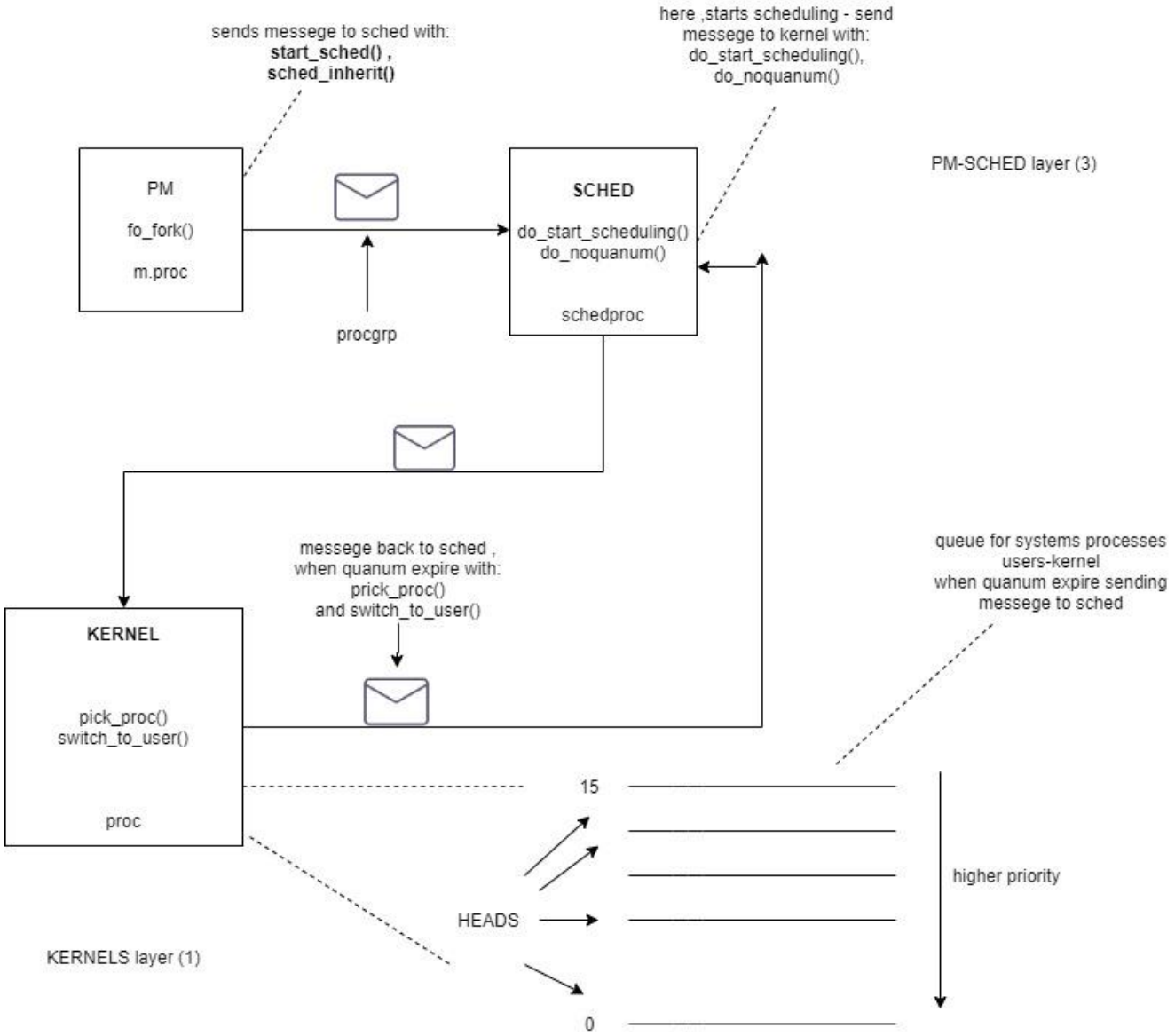
ΠΕΡΙΕΧΟΜΕΝΑ

ΠΕΡΙΛΗΨΗ ΑΝΑΦΟΡΑΣ ΚΑΙ ΣΗΜΕΙΟ ΥΛΟΠΟΙΗΣΗΣ

Για την δεύτερη εργαστηριακή άσκηση , θα σας παρουσιάσουμε την δική μας λύση μέχρι και το τρίτο ερώτημα φτάνοντας σε ένα σημείο από αυτό.Ολοκληρωσαμε τις αλλαγες στην ουρα και στην συνεχεια το μονο που καταφεραμε ηταν να στειλουμε (διαπιστωνοντας το με printf) το fss_priority στον πυρηνα από τον sched με την μορφη μηνυματος,οποτε δεν εγινε η συνεχεια του τριτου ερωτηματος και προφανως το τεταρτο .Κύριο εργαλείο μας στην μεταγλώτιση του προγράμματος ήτανε οι εντολές (**make world**) στον φάκελο **/usr/src/** ή επίσης και οι εντολές **make libraries – make includes** και στον φάκελο **/usr/src/tools make install.** Η παρακάτω αναφορά περιλαμβάνει αναλυτική εξήγηση για τις αλλαγές στον κώδικα μέχρι το σημείο που καταφέραμε να υλοποιήσουμε .Επίσης θα σας παρουσιάσουμε και ένα σχετικό διάγραμμα επικοινωνίας των servers από τον PM – SCHED – KERNEL.

ΣΗΜΕΙΩΣΗ **:** Για τις πρώτες κινήσεις μας στον κώδικα μας βοήθησαν αρκετά οι οδηγίες του κυρίου Καππέ στο σχετικό φροντιστήριο.

ΕΠΙΚΟΙΝΩΝΙΑ PM –SCHED -KERNEL



ΥΛΟΠΟΙΗΣΗ ΠΡΩΤΟΥ ΕΡΩΤΗΜΑΤΟΣ

- Παρατηρώντας την main συνάρτηση του PM βλέπουμε πως στο σημείο που εξετάζει τους τύπους μηνυμάτων που δέχετε (swich(call_nr)) καλεί σε ένα case από αυτά την συνάρτηση **sched_start_user() .** Αυτό συμβαίνει για τον λόγο ότι : μόλις κληθεί η do_fork() και έπειτα από μια ακολουθεία επικοινωνίας μέσω μηνυμάτων μεταξύ των servers ο PM θα λάβει τύπο μηνύματος από τον VFS PM_FORK_RYPLE εκεί γίνεται η κλίση της παραπάνω συνάρτησης , η οποία είναι και η εναρκτήρια για την επικοινωνία των PM-SCHED.
    1. Πηγαίνουμε στην /usr/src/servers/pm/schedule.c βρίσκουμε την συναρτηση που μας ενδιαφέρει(sched_start_user()) και παρατηρούμε ότι στο τέλος καλεί την συνάρτηση sched_inherit().Εκεί δίνει σαν ορίσματα καποια πεδία τα οποία είναι απαραίτητα τόσο για τις διεργασίες όσο και για τον sched ώστε να μπορέσει να κανει την δουλεία του .Εκεί λοιπόν θα τοποθετήσουμε το πεδίο procgrp ώστε να κάνουμε μια αρχή για να μπορέσει να το δει ο sched .

Οπότε παμε στην σειρά 83 και το τοποθετούμε ως εξής :

return sched_inherit(ep,rmp->mp_endpoint, inherit_from,maxprio,&rmp->mp_scheduler ,**rmp->mp_procgrp**);

Την συγκεκριμένη ενέργεια μπορούμε να την κάνουμε καθώς αν πάμε στο header file **/usr/src/servers/pm/mproc.h** παρατηρούμε στο struct mproc ότι υπάρχει το πεδίο σε αυτή την μορφή .

Κάνοντας compile το σύστημα υπάρχουν erros καθώς δεν ορίσαμε το πεδίο που βάλαμε στην sched_inherit() οπότε εκείνη βλέπει ένα παραπάνω πεδίο έτσι:

2. πάμε στην /usr/src/lib/libsys/sched_start.c στο αρχείο αυτό βρίσκουμε την συνάρτηση sched_inherit() και βαζούμε ένα ακόμη πεδίο στην δήλωσή της ως εξής :

PUBLIC int sched_inherit(endpoint_t scheduler_e, endpoint_t schedulee_e, endpoint_t parent_e, unsigned maxprio,endpoint_t *newscheduler_e, **pid_t procgrp**)

2.1. Για να υπάρχει όμως λόγος ύπαρξης του συγκεκριμένου πεδίου

Πάμε στο **/usr/src/include/minix/sched.h** εντοπίζουμε την δήλωση της συνάρτησης και της βάζουμε ακόμη ένα όρισμα το procgrp

Στο σημείο αυτό τώρα θα πρέπει να αρχικοποιηθεί το πεδίο που προσθέσαμε στην sched_inherit() όπως και τα άλλα.Για να γίνει αυτό χρειάζεται να γνωρίζουμε τον τύπο μηνύματος του πεδίου. Τα messeges αυτά στην συγκεκριμένη συνάρτηση είναι τύπου scheduling messeges , άρα επικεντρώνουμε την αναζήτηση μας σε αυτού του ήδους μηνύματα. Παρατηρώντας στον φάκελο που βρίσκονται οι τύποι μηνυμάτων /usr/src/include/minix στο αρχείο com.h υπάρχουν στην σειρά 1147 δηλώσεις μεσω #define messeges for scheduling.Εκεί παρατηρείται ότι στα defines από κάτω υπαρχουν τα πεδία που χρησιμοποιεί η inherit() ως μηνύματα τύπου m9 .Άρα και το δικό μας θα είναι τετοιου τύπου πηγαινοντας στον ιδιο φακελο αλλα στο αρχειο ipc.h παρατηρουμε ότι ο τυπος m9_l5 είναι κενος αρα παμε στο com.h και κανουμε define ως εξης : #define SCHEDULING_TEAMNUM m9_l5.Κάνοντας το συγκεκριμένο βήμα επιστρέφουμε στο /usr/src/lib/libsys/sched_start.c ώστε μεσα στην συναρτηση inherit() να αρχικοποιήσουμε το πεδιο progrp στην σειρα 31

m.SCHEDULING_TEAMNUM = procgrp;

3. για να μπορέσει ο sched να δει το procgrp προσθετουμε στο struct από το header file του το συγκεκριμενο πεδιο , αρα:

usr/src/servers/sched/schedproc.h μεσα στο struct το αρχικοποιούμε ως

pid_t procgrp;

4. Η main συνάρτηση του sched εχει και αυτή καποια cases τα οποια είναι για τους τυπους μηνυμάτων που λαμβανει . Όταν δεχθει το μηνυμα από τον PM μέσω της sched_inherit() τότε καλείται η do_start_scheduling() ώστε να ξεκινήσει η χρονοδρομολόγηση .Εκει λοιπόν θα αρχικοποιήσουμε το μήνυμα στον sched δηλαδή τον οδηγό ομαδος (procgrp) ετσι ώστε να το εχει και εκεινος μέσο του ορίσματος m_ptr. Αρα /usr/src/servers/sched/schedule.c στην συναρτηση do start_scheduling() και σειρα 167

rmp->procgrp=m_ptr->SCHEDULING_TEAMNUM

ΥΛΟΠΟΙΗΣΗ ΔΕΥΤΕΡΟΥ ΕΡΩΤΗΜΑΤΟΣ

Αρχικά , όπως μας αναφερεται και στο ερωτημα 2 τροποποιησαμε το struct schedproc τοποθετώντας μεσα αλλα τρια σημαντικα πεδία για τις διεργασιες.Οπότε παμε στο /usr/src/servers/sched/schedproc.h και τοποθετουμε τα πεδια proc_usage , grp_usage και fss_priority ως pid_t . Για την δικαιη χρονοδρομολογηση των διεργασιων αυτά τα τρια πεδια είναι πολύ σημαντικα μαζι και με το procgrp, το οποιο είναι ηδη αρχικοποιημενο στον χρονοδρομολογητη sched καθως το συμπεριλαβαμε στο μηνυμα που του στελνει ο PM servers συμφωνα με οσα ειπαμε στην υλοποιηση του πρωτου ερωτηματος .Ολες οι αλλαγες που εγιναν πανω στον κωδικα για το συγκεκριμενο ερωτημα είναι στον φακελο /usr/src/servers/sched και συγκεκριμενα στο αρχειο schedule.c .Αρχικα ξεκινησαμε με την αρχικοποιηση των πεδιων στην do_start_scheduling(), το proc_usage τ αφησαμε ισο με 0 καθως δεν εχει λαβει επεξεργαστη ακομη . Για το grp_usage , τρεχουμε μεσα σε ένα loop να βρουμε μια διεργασια στο ιδιο group , δηλαδη με ιδιο αριθμο procgrp , ώστε να αντιγραψουμε το grp_usage αυτης στο πεδιο που θελουμε να αρχικοποιησουμε .Οσο αναφορα το fss_priority υλοποιησαμε μια συναρτηση η οποια υπολογιζει τον αριθμο των groups. Στην συναρτηση αυτή δεν χρησιμοποιηθηκε μεθοδος ταξινομησης , όμως υλοποιηθηκε ως εξης : Αφου ελεγξουμε πρωτα αν είναι διεργασια χρηστη (rmp->priority==USER_Q) και τρεχει στον sched εκεινη την στιγμη , αποθηκευσαμε μεσα σε έναν πινακα τους αριθμους procgrp κάθε διεργασιας .

Στο σημειο αυτό καλο θα ητανε να αναφερουμε ότι , την στιγμη που ξεκινα την λειτουργια του το επιπεδο 3 των servers , η πρωτη διεργασια που δημιουργειται και τρεχει συνεχως είναι η init. Εκεινη είναι υπευθυνη να δημιουργει τις διεργασιες χρηστη και να τις ελεγχει .Απο την στιγμη που ξεκινα μια init θα δωσει σε οσες διεργασιες δημιουργησει τον οδηγο ομαδος της (procgrp).Ετσι λοιπον συνεχιζοντας στην υλοποιηση την στιγμη που προσθεσουμε στον πινακα τους procgrp , αυξανουμε και έναν counter (bind) ώστε να γνωριζουμε ποσες θεσεις από τις 256 του NR_PROCS χρησιμοποιησε .Επειτα τρεχοντας τον πινακα απαλοιφουμε πιθανες διπλοτιμες που μπορει να εχουν αποθηκευτει στα κελια από την παραπανω διαδικασια.Το γεγονος αυτό γινεται για να μην αυξανεται ο counter μας για τα groups όταν βλεπει ιδια τιμη procgrp, δλδ να αλλαζει μονο για διαφορετικες τιμες procgrp .Ετσι όταν το κελι του πινακα δεν είναι μηδεν αυξανεται ο counter των groups . Οποτε συμφωνα και με τον τυπο που οριζεται στην εκφωνηση αρχικοποιουμε το fss_priority .Τωρα όταν μια διεργασια τελειωνει το κβαντο της η συναρτηση που αναλαμβανει να διαχειριστει την αμεσως επομη κινηση της είναι η do_noquanum().Εκει κανουμε την ενημερωση των πεδιων συμφωνα με οσα οριζονται στο μαθημα αλλα και στην εκφωνηση .

ΣΧΟΛΙΑ ΕΡΩΤΗΜΑΤΟΣ 2

1. Στον κωδικα που θα σας παρουσιασουμε στην συνεχεια , στο αρχειο schedule.c του φακελου /usr/src/servers/sched/ τοσο στην αρχικοποιηση των πεδιων οσο και στην ενημερωση τους θα μπορουσαμε να βαλουμε ελεγχο αν το priority==USER_Q , όμως δεν είναι "απαραίτητο" διοτι το priority είναι συνεχως στην τιμη 7 , αλλα επισης στην συνεχεια αλλαζουμε τις τιμες των MIN_USER_Q = MAX_USER_Q = USER_Q =7 , οποτε γι αυτόν τον λογο δεν χρησιμοποιηθηκε ο ελεγχος.
2. Στην ενημερωση και αρχικοποιηση των πεδιων αλλα και στην ευρεση των groups χρησιμοποιήθηκε η εντολη rmp->flags & IN_USE , ώστε να γνωριζουμε αλλα και να χρησιμοποιουμε δεδομενα από διεργασιες που τρεχουν εκεινη την στιγμη στον schedproc
3. Οι διεργασιες του πυρηνα εχουν μεγαλυτερη προτεραιοτητα από τις διεργασιες χρηστη .

ΥΛΟΠΟΙΗΣΗ ΤΡΙΤΟΥ ΕΡΩΤΗΜΑΤΟΣ

- Η υλοποίηση αυτου του ερωτηματος βασιζεται στην επικοινωνια του πυρηνα (kernel) αλλα και του sched!Στο σημειο αυτό καλο θα ητανε να αναφερθουμε στο πως επιτυγχανεται αυτή η επικοινωνια .

1. ΑΠΌ ΠΛΕΥΡΑ SCHED

Στο αρχειο /usr/src/servers/sched/schedule.c παρατηρουμε πως όταν μια διεργασια τελειωσει το κβαντο της και μπαινει στην συναρτηση do_noquanum() , εκεινη καλει στο τελος μια άλλη συναρτηση που λεγεται sys_schedule_local().Αυτη ουσιαστικα είναι η schedule_process(),οπου δημιουργει τα καταλληλα πεδια για την αποστολη προς τον kernel τα οποια τα στελνει μεσω της sys_schedule() στον kernel .Εμβαθύνοντας τωρα αν παμε στον φακελο και συγκεκριμενα στο αρχειο που υλοποιειται η sys_schedule() /usr/src/lib/libsys/sys_schedule.c , βλεπουμε πως η δουλεια της είναι να αρχικοποιει τα πεδια που της δινονται και να δημιουργει το μηνυμα που θα σταλθει στον kernel .Στο τελος καλει την συναρτηση kernel_call() και ξεκινα η διαδικασια επικοινωνιας .

2. **ΑΠΌ ΤΗΝ ΠΛΕΥΡΑ ΤΟΥ KERNEL** τωρα καλο θα ητανε να δουμε πως λαμβανει αυτό το μηνυμα .Πηγαινοντας στον φακελο /usr/src/kernel/system/ και στο αρχειο do_schedule.c που μας ενδιαφερει για την χρονοδρομολογηση , παρατηρειται πως λαμβανει το συγκεκριμενο μηνυμα από την kernel_call() και ανακτα τις τιμες των πεδιων . Τελος καλειται η συναρτηση sched_proc().

- Τωρα για την υλοποιηση του τριτου ερωτηματος μειωσαμε τις ουρες από 15 σε 8 οπου στην θεση 8 βρισκεται η idle η οποια περιμενει στο συστημα και δεν κανει τιποτα . Οι τιμες των max-min και userq τοποθετηθηκαν στην τιμη 7 ώστε ολες οι διεργασιες χρηστη να βρισκονται στην συγκεκριμενη θεση .Το πρωτο μας βημα σε αυτό το σημειο ηταν να στειλουμε την τιμη fss_priority στον πυρηνα μεσο της διαδικασιας που εξηγηθηκε παραπανω.Ετσι λοιπον :

1. /usr/src/servers/sched/schedule.c τοποθετησαμε το πεδιο fss_priority

Στην sys_schedule(..,rmp->fss_priority)

2. /usr/src/lib/libsys/sys_schedule.c εκει προσθεσαμε το πεδιον συναρτηση κι επειτα ολοκληρωσαμε την αρχικοποιηση του ως εξης :

m.SCHEDULING_FSSPRIO=priority , αφου πρωτα μεταβουμε στο header αρχειο

3. /usr/src/include/minix/com.h ώστε να ορισουμε τον τυπο μηνυματος του συγκεκριμενου πεδιου κι επειτα

Σημειωση : Γνωριζουμε ότι μπορει να μην εχει νοημα καθως δεν υλοποιησαμε την συνεχεια . Αυτό που σκεφτηκαμε να κανουμε είναι όταν στην do_schedule()

Στον φακελο /usr/src/kernel αρχικοποιησαμε το πεδιο για να μπορεσει ο kernel να επιλεγει το μικροτερο fss_priority θα επρεπε να το πεδιο αυτό να εμφανιστει μεσα στην pick_proc() στον φακελο /usr/src/kernel/proc.c . Η συγκεκριμενη συναρτηση εχει ένα loop που διαχειριζεται την ουρα στην οποια αποθηκευονται οι διεργασιες εκει θα επρεπε να επιλεγεται το μικροτερο fss_priority . Για να φτασει όμως εκει το πρωτο βημα θα ητανε να προσθεταμε το fss_priority στην

Sched_proc(…,fss_priority) σαν πεδιο στο /usr/src/kernel/do_schedule().Η υλοποιηση αυτή δεν εγινε καθως δυσκολευτηκαμε αρκετα για το πως θα ενημερωνουμε τον πυρηνα από την do_noquanum() βαζοντας σε ένα loop την sys_schedule() , όπως ειχε αναφερθει και στο φροντηστηριο του κυριου Καππέ!

ΚΩΔΙΚΑΣ ΑΛΛΓΕΣ

1. /usr/src/lib/libsys

Sched_start.c

Σειρα (14):PUBLIC int sched_inherit(endpoint_t scheduler_e, endpoint_t schedulee_e, endpoint_t parent_e, unsigned maxprio, endpoint_t *newscheduler_e,**pid_t procgrp**)

Σειρα (31) : **m.SCHEDULING_TEAMNUM =procgrp;**

Sys_schedule.c

Σειρα (3) : PUBLIC int sys_schedule(endpoint_t proc_ep, int priority, int quantum, int cpu,**pid_t fss_priority**)

Σειρα (14) : **m.SCHEDULING_FSSPRIO = fss_priority;**

2. /usr/src/servers/sched

Schedproc.h

Σειρα (28-31):**pid_t procgrp;**

pid_t proc_usage;

pid_t grp_usage;

pid_t fss_priority;

Schedule.c

Σειρα (87-128):

```
PUBLIC int groupClassification(){
   struct schedproc *nik; //tha trekso ton sched gia na bro diergasia pou idi trexei
    int grpArray[NR_PROCS]={0};
    int block,i,j;
    int number_of_groups=0;

    int bind=0;
    for(i=0,nik=schedproc;i<NR_PROCS;i++,nik++){
       if(nik->procgrp>10){
                    grpArray[bind]=nik->procgrp;
                    bind++;
       }

    }
```

```
    for(i=0;i<bind;i++){
      for(j=i+1;j<bind;j++){
         if(grpArray[i]==grpArray[j]){
            for(int k=j;k<bind;k++){
               grpArray[k]=grpArray[k+1];
            }
            bind=bind-1;
            j=j-1;
         }
      }
   }
   for(i=0;i<bind;i++){
      if(grpArray[i]!=0){
         number_of_groups =number_of_groups+1;
      }
   }


    return number_of_groups;
}
```

συναρτηση **do_noquantum()**

Σειρα(137-138): **struct schedproc *scd2;**

int i=0,base=0;

Σειρα (150-160):

```
rmp->proc_usage = rmp->proc_usage + 200;
   rmp->grp_usage  = rmp->grp_usage  + 200;
   for(i=0,scd2=schedproc;i<NR_PROCS;i++,scd2++){
      if(scd2->flags & IN_USE && scd2->priority==USER_Q ){
         scd2->proc_usage=scd2->proc_usage/2;
         scd2->grp_usage=scd2->grp_usage/2;
         scd2->fss_priority=scd2->proc_usage/2+scd2-
  >grp_usage*groupClassification()/4+base;


      }
   }
   rmp->fss_priority=rmp->proc_usage/2+rmp-
>grp_usage*groupClassification()/4+base;
```

Συναρτηση **do_start_scheduling()**

Σειρα(210): **struct schedproc *scd;**

Σειρα (231): **rmp->procgrp = m_ptr->SCHEDULING_TEAMNUM;**

Σειρα(317-319):**rmp->proc_usage = 0;**
rmp->fss_priority=0;
rmp->grp_usage = 0;

Σειρα(323-332):

```
int i;
for(i=0,scd=schedproc;i<NR_PROCS;i++,scd++){
   if(scd->flags & IN_USE){
      if(scd->procgrp == rmp->procgrp){
         rmp->grp_usage=scd->grp_usage;
      }
   }
}
int base=0;
   rmp->fss_priority=rmp->proc_usage/2+rmp->grp_usage*groupClassification()/4+base;
```

Σειρα(418): συναρτηση **schedule_process()**

```
if ((err = sys_schedule(rmp->endpoint, new_prio,
        new_quantum, new_cpu,rmp->fss_priority)) != OK) {
        printf("PM: An error occurred when trying to schedule %d: %d\n",
        rmp->endpoint, err);
    }
```

3. /usr/src/include/minix

Com.h

Σειρα(1129-1128):

```
#      define SCHEDULING_TEAMNUM      m9_l5
#    define SCHEDULING_FSSPRIO    m9_s4
```

Sched.h

```
_PROTOTYPE(int sched_inherit, (endpoint_t scheduler_e,
    endpoint_t schedulee_e, endpoint_t parent_e, unsigned maxprio,
    endpoint_t *newscheduler_e,pid_t procgrp));
```

3. /usr/src/servers/pm

Schedule.c

Σειρα(79):

```
 return sched_inherit(ep,          /* scheduler_e */
      rmp->mp_endpoint,          /* schedulee_e */
      inherit_from,              /* parent_e */
      maxprio,                   /* maxprio */
      &rmp->mp_scheduler,rmp->mp_procgrp);
```