Planificador Round Robín (Clásico)

Void Planificador () {

If (Finalizo (PRUN))

FreeMen (PRUN);

else [

PRUN.Reg=CPU.Reg;

Q.Meter(PRUN);

]

PRUN=Q.Sacar();

CPU.Reg=PRUN.Reg;

}

* Escriba el código del RR de modo que asigne 2 quantum por proceso

int cp=0;

void Planificador(){

cp++;

if(cp==2 || Finalizo(PRUN)){

if(Finalizo(PRUN))

FreeMen(PRUN);

else[

PRUN.Reg=CPU.Reg;

Q.Meter(PRUN);

]

PRUN=Q.Sacar();

CPU.Reg=PRUN.Reg;

cp=0;

}

}

Planificador por Prioridad (Clásico)

void Planificador(){

if(Finalizo(PRUN))

FreeMen(PRUN);

else[

PRUN.Reg=CPU.Reg;

int i=PRUN.Prioridad;

Q[i].Meter(PRUN);

]

.

.

.

}

* Escribir un planificador para N colas de prioridad Q[i],Q[2],…,Q[N] el cual asigne K quantum por cola a la Q[K]

int cqc=0; int K=N;

void Planificador(){

if(Finalizo(PRUN))

FreeMen(PRUN);

else[

PRUN.Reg=CPU.Reg;

int i=PRUN.Prioridad;

Q[i].Meter(PRUN);

]

Q[K].Sacar();

cqc++;

if(cqc==K){

cqc=0;

k--;

if(k==0)

k=N;

}

CPU.Reg=PRUN.Reg;

}

* Resolver le planificador anterior que asigne X quantums por cola a la Q[X]

Y 2 quantum por proceso.

int cqc=0; int K=N; int cqp=0;

void Planificador(){

cqp++;

if(cqp==2 || Finalzo(PRUN)){

if(Finalizo(PRUN))

FreeMen(PRUN);

else[

PRUN.Reg=CPU.Reg;

int i=PRUN.Prioridad;

Q[i].Meter(PRUN);

]

Q[K].Sacar();

cqc++;

if(cqc==K){

cqc=0;

k--;

if(k==0)

k=N;

}

CPU.Reg=PRUN.Reg;

cqp=0;

}

}

* Escribir el código de un planificador RR el cual asigne 2 quantum por proceso a los PID par y 3 quantum por proceso a los de PID Impar.

int cp=0; int tope=0;

void Planificador(){

if(PRUN.PID%2==0)

tope=2;

tope=3;

cp++;

if(cp==tope || Finalizo(PRUN)){

if(Finalizo(PRUN))

FreeMen(PRUN);

else[

PRUN.Reg=CPU.Reg;

Q.Meter(PRUN);

]

PRUN=Q.Sacar();

CPU.Reg=PRUN.Reg;

cp=0;

}

}

* En un SO corren procesos que nunca finalizan por sus propios medios. Pero el planificador los finaliza –les hace FreeMen- después que un proceso a recibido 50 quantums.

/\*Se asume que el PCB tiene un campo adicional para contar los quantums “Cantq” \*/

void planificador(){

Contq++;

if(Contq==50)

FreeMen(PRUN)

else{

PRUN.Reg=CPU.Reg;

Q.Meter(PRUN);

}

PRUN=Q.Sacar();

CPU.Reg=PRUN.Reg;

}

* Escribir un planificador con N colas de prioridad Q[1]…..Q[N]. el cual asigna 1 quantum por cola a las colas impares y 2q por cola a las impares. (en todos los casos se asigna 1q por proceso).

Ver 1

int k=N; int cqc=0; int tope=0;

void planificador(){

if(Finalizo(PRUN))

FreeMen(PRUN);

else{

PRUN.Reg=CPU.Reg;

int i=PRUN.prioridad;

Q[i].Meter(PRUN);

}

PRUN=Q[K].Sacar();

cqc++;

if(PRUN.Prioridad%2==0)

tope=2;

tope=1;

if(cqc==tope)

k--;

if(k==0)

k=N;

cqc=0;

CPU.Reg=PRUN.Reg;

}

Ver 2

int ProximaCola(int k){

if(k==0)

k=N;

return K--;

}

int QPorCola(int k){

if(k%2==0)

return 2;

return 1;

}

int k=N; int cqc=0; int tope=0;

void planificador(){

if(Finalizo(PRUN))

FreeMen(PRUN);

else{

PRUN.Reg=CPU.Reg;

int i=PRUN.prioridad;

Q[i].Meter(PRUN);

}

PRUN=Q[K].Sacar();

cqc++;

if(cqc==QPorCola(k)){

k=ProximaCola(k);

cqc=0;

}

CPU.Reg=PRUN.Reg;

}

* Utilizando el esquema anterior escribe un planificador con 3 colas de prioridad asignando 1q por proceso y:
  + Q[1] 7q por cola
  + Q[2] 3q por cola
  + Q[3] 1q por cola

int ProximaCola(int k){

if(k==0)

k=N;

return K--;

}

int QPorCola(int k){

if(k==1)

return 7;

else{

if(k==2)

return 3

}

Return 1

}

int k=N; int cqc=0; int tope=0;

void planificador(){

if(Finalizo(PRUN))

FreeMen(PRUN);

else{

PRUN.Reg=CPU.Reg;

int i=PRUN.prioridad;

Q[i].Meter(PRUN);

}

PRUN=Q[K].Sacar();

cqc++;

if(cqc==QPorCola(k)){

k=ProximaCola(k);

cqc=0;

}

CPU.Reg=PRUN.Reg;

}

* Reescribe el ejercicio anterior pero asignando 2q por proceso a las colas impares y 1q por proceso a las colas pares.

int ProximaCola(int k){

if(k==0)

k=N;

return K--;

}

int QPorCola(int k){

if(k%2==0)

return 2;

return 1;

}

int QPorProc(PCB PRUN){

if(PRUN.Prioridad%2==0)

return 1;

return 2;

}

int k=N; int cqc=0; int cqp=0

void planificador(){

cqp++;

if(cqp==QPorProc(PRUN)|| Finalizo(PRUN))

if(Finalizo(PRUN))

FreeMen(PRUN);

else{

PRUN.Reg=CPU.Reg;

int i=PRUN.prioridad;

Q[i].Meter(PRUN);

}

PRUN=Q[K].Sacar();

cqc++;

if(cqc==QPorCola(k)){

k=ProximaCola(k);

cqc=0;

}

CPU.Reg=PRUN.Reg;

}

}

* Escribir un planificador con N colas de prioridad q asigne Xq por cola Q[X] y Xq por proceso.

int cqc=0; int K=N; int cqp=0;

void Planificador(){

cqp++;

if(cqp==k || Finalizo(PRUN)){

if(Finalizo(PRUN))

FreeMen(PRUN);

else[

PRUN.Reg=CPU.Reg;

int i=PRUN.Prioridad;

Q[i].Meter(PRUN);

]

Q[K].Sacar();

cqc++;

if(cqc==K){

cqc=0;

k--;

if(k==0)

k=N;

}

CPU.Reg=PRUN.Reg;

cqp=0;

}

}

* Escribir un planificador RR para q se elimine (FreeMen) el proceso que tenga el mayor PID de la RAM. (Solo eliminar 1). Se cuanta con un variable global N para el numero de procesos

Ver 1

int i=1; /\*Var N\*/ int May=0;

void planificador(){

if(i<=N){

if(PRUN.PID>May)

May=PRUN.PID;

PRUN.Reg=CPU.Reg;

Q.Meter(PRUN);

PRUN.Sacar();

CRP.Reg=PRUN.Reg;

i++;

}

if(PRUN.PID==May)

FreeMen(PRUN)

else{

PRUN.Reg=CPU.Reg;

Q.Meter(PRUN);

}

PRUN=Q.Sacar();

CRP.Reg=PRUN.Reg;

}

Ver 2

int may=0;

void planificador(){

if(PRUN.PID==May)

FreeMen(PRUN);

else{

if(PRUN.PID>May)

May=PRUN.PID;

PRUN.Reg=CPU.Reg;

Q.Meter(PRUN);

}

PRUN=Q.Sacar();

CRP.Reg=PRUN.Reg;

}

* Escribir un planificador RR el cual asigne 1q x proceso a lo que entran por primera vez, 2q por proceso a los q entran por 2da Vez y asi sucesivamente. (Puedes inventar y utilizar un Campo en el PCB del PRUN para ayudarte)

tope=0; cp=0;

void planificador(){

cp++;

tope=PRUN.Cant;

if(cp==tope || Finalizo(PRUN)){

cp=0;

PRUN.Cant++;

if(Finalizo(PRUN)){

FreeMen(PRUN);

}

else{

PRUN.Reg=CPU.Reg

Q.Meter(PRUN)

}

PRUN=Q.Sacar();

CRP.Reg=PRUN.Reg;

}

}

* Escribir un planificador con N colas de prioridad que asigne X quantums por Q[X], y si el PID del proceso es Par se lo coloca en la cola Siguiente.

int cqc=0;

void planificador(){

if(Finalizo(PRUN))

FreeMen(PRUN);

else{

PRUN.Reg=CPU.Reg

if(PRUN.PID%2==0){

k--

if(k==0)

Q[N].Meter(PRUN);

Q[k-1].Meter(PRUN);

}

else{

Q[K].Meter(PRUN);

}

cqc++;

if(cqc==k){

k--

cqc=0;

}

if(k==0)

k=N

PRUN=Q{k}.Sacar();

CPU.Reg=PRUN.Reg;

}

}

* Escribir un planificador con cola de prioridad para N colas que asigne X quantum por proceso Si se encuentra en su cola de origen (Q[K]=PRUN.Prioridad) y 1q por proceso si no. Además los procesos debes meterse en la cola siguiente.

int cqp=0;

void planificador(){

cqp++;

if(PRUN.prioridad==k)

tope=3;

tope=1;

if(cqp==tope|| Finalizo(PRUN)){

if(Finalizo(PRUN))

FreeMen(PRUN)

else{

PRUN.Reg=CPU.Reg;

k--;

if(k==0)

k=N;

Q[k].Meter(PRUN);

}

PRUN=Q[k].Sacar();

CPU.Reg=PRUN.Reg;

cqp=0;

}

}