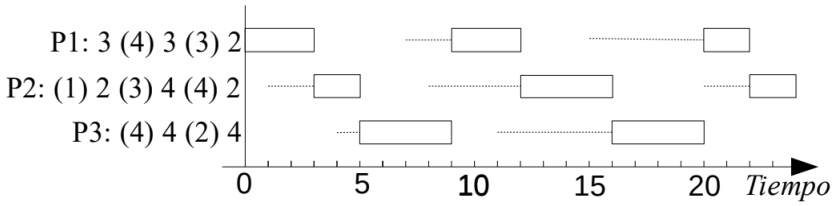
CC4302 – Sistemas Operativos
Pauta Auxiliar 6

Profesor: Luis Mateu
Auxiliar: Diego Madariaga

29 de abril de 2020

1. P1.b Control 2 2019/2

Se trata de scheduling con prioridades



2. Impresora Server

```
/*** implementacion ***/
#define OBTENER 1
#define DEVOLVER 2
#define OK 3

// Esta es la tarea que administra la impresora
nTask impresora;

void obtenerImpresora(){
    // Enviar el mensaje de "Obtener" la impresora
    int msg = OBTENER;
    nSend(impresora, &msg);
}

void devolverImpresora(){
    // Enviar el mensaje de "Devolver" la impresora
    int msg = DEVOLVER;
    nSend(impresora, &msg);
}

int impresoraServer(){
    nSetTaskName("TASK impresoraServer");
    nTask t;
    int *msg;
    // Cola de trabajos que deben esperar
    FifoQueue printerQueue = MakeFifoQueue();
    int busy = FALSE;
```

```
    for(;;) {
        if(busy == FALSE){
            // Recibir mensajes por cierto tiempo
            msg = (int *) nReceive(&t, MILISEGUNDOS_DORMIR);
            if (t == NULL) {
                // Se cumplio el tiempo sin recibir mensajes, pasar a bajo consumo
                modoBajoConsumo();
                // Esperar indefinidamente por un trabajo
                msg = (int *) nReceive(&t, -1);
                // Despertar la impresora cuando estaba dormida
                modoUsoNormal();
            }
        } else {
            // Si la impresora esta ocupada, esperar indefinidamente
            // por el mensaje "Devolver"
            msg = (int *) nReceive(&t, -1);
        }
        if(*msg == OBTENER){
            if(busy == TRUE){
                // La impresora esta ocupada, agrego la solicitud a la fila
                PutObj(printerQueue, t);
            } else {
                // Ocupar la impresora
                busy = TRUE;
                nReply(t, OK);
            }
        } else if(*msg == DEVOLVER){
            // Responder a la tarea
            nReply(t, OK);
            // Ver si hay otra tarea esperando por la impresora
            if (!EmptyFifoQueue(printerQueue)){
                nTask *t2 = (nTask *) GetObj(printerQueue);
                nReply(t2, OK);
            } else {
                // Si no hay otra tarea, la impresora queda disponible
                busy = FALSE;
            }
        }
    }
}

void inicializarImpresora(){
    // Este procedimiento solo lanza la tarea con el printServer
    impresora = nEmitTask(impresoraServer);
}
```

3. Mensajería en nSystem

```
int nSend(nTask task, void *msg) {
        int rc;
        START_CRITICAL(); //deshabilita interrupciones
        nTask this_task= current_task;
        if (task->status==WAIT_SEND) {
                task->status= READY;
                /* En primer lugar en la cola */
                PushTask(ready_queue, task);
        }
        PutTask(task->send_queue, this_task);
        this_task->send.msg= msg;
        this_task->status= WAIT_REPLY;
        ResumeNextReadyTask(); //cambio de contexto
        rc= this_task->send.rc;
        END_CRITICAL();
        return rc;
```

```
}

void *nReceive(nTask *ptask, int timeout) {
        void *msg;
        nTask send_task;
        START_CRITICAL();
        nTask this_task= current_task;
        if (EmptyQueue(this_task->send_queue) && timeout != 0) {
                /* La tarea espera indefinidamente */
                this_task->status= WAIT_SEND;
                /* Se suspende indefinidamente hasta un nSend */
                ResumeNextReadyTask();
        }
        send_task= GetTask(this_task->send_queue);
        if (ptask!=NULL) *ptask= send_task;
        msg= send_task==NULL ? NULL : send_task->send.msg;
        END_CRITICAL();
        return msg;
}

void nReply(nTask task, int rc) {
        START_CRITICAL();
        PushTask(ready_queue, current_task);
        task->send.rc= rc;
        task->status= READY;
        PushTask(ready_queue, task);
        ResumeNextReadyTask();
        END_CRITICAL();
}
```