

28 Sep 00 14:57

producteur-consommateur.c

Page 1/3

```

#include "PRELUDE.h"

#define DEBUG 1

#define MAX_ELEMS 100000

typedef int *GLOBAL int_GPTR; /* Adresse globale d'une variable int. */
typedef SPTR *GLOBAL SPTR_GPTR; /* Adresse globale d'une variable SPTR. */

/* Macro OUT: sert a indiquer de facon claire et explicite qu'un
   argument sert uniquement a retourner un resultat.
   Par default, tous les autres arguments sont IN.*/
#define OUT *GLOBAL

THREADED
producteur(
    /* Nombre d'elements a produire et copier. */
    int nb_elements,
    /* Tampon ou copier les caracteres. */
    int_GPTR buf_ref,
    /* Sync slot a signaler lorsque le buffer est libre. */
    SPTR OUT buf_libre,
    /* Adresse de la variable qui contiendra la sync slot (dans
       le consommateur) que le producteur signalera lorsque le
       tampon sera plein. */
    SPTR_GPTR OUT buf_plein_ref,
    /* Slot a signaler lorsque l'initialisation est terminee. */
    SPTR fin_init,
    /* Slot a signaler lorsque tous les elements ont ete produits. */
    SPTR fin
)
{
    int a[MAX_ELEMS]; /* Le tableau a copier. */
    int p; /* Position du prochain element a copier. */
    SPTR buf_plein; /* Slot (dans le consommateur) a signaler lorsque le
                    tampon est plein. */

    /* On transmet la sync slot pour buf_libre et l'adresse de la
       variable locale pour buf_plein au processus maitre. */
    PUT_SYNC( TO_SPTR(COPIER_ITEM), buf_libre, fin_init );
    PUT_SYNC( TO_GLOBAL(&buf_plein), buf_plein_ref, fin_init );

    /* On cree un tableau (pour tests) et on initialise le pointeur
       d'element a copier. */
    for( p = 0; p < nb_elements; p++ )
        a[p] = 10*(p + 1);
    p = 0;

    /* La fibre qui fait le travail de copier un element apres l'autre. */
    FIBER COPIER_ITEM <* 1 *> {
        if (DEBUG) printf( "producteur[COPIER_ITEM]: p = %d\n", p );
        if (p < nb_elements) {
            /* Il reste au moins un element a transmettre. */
            PUT_SYNC( a[p], buf_ref, buf_plein );
            p += 1;
        } else {
            /* Dernier element transmis: on signale la terminaison. */
            SYNC( fin );
            TERMINATE;
        }
    }
}
}
/*

```

```

*/
THREADED
consommateur(
    /* Adresse de la variable qui sert de tampon (buffer). */
    int_GPTR OUT buf_ref,
    /* Sync slot a signaler lorsque le tampon est plein. */
    SPTR OUT buf_plein,
    /* Adresse de la variable qui contiendra la sync slot
       (dans le producteur) que le consommateur signalera
       lorsque le tempon sera libre. */
    SPTR_GPTR OUT buf_libre_ref,
    /* Sync slot pour terminer l'execution du consommateur. */
    SPTR OUT terminer_consommateur,
    /* Slot a signaler lorsque l'initialisation est terminee. */
    SPTR fin_init,
    /* Slot a signaler lorsque tous les elements ont ete recus
       et traites (c'est-a-dire imprimes). */
    SPTR fin
)
{
    int b[MAX_ELEMS]; /* Le tableau ou les elements seront copies. */
    int buf;          /* Le tampon dans lequel le producteur
                       copiera les elements. */

    int c;           /* Position du prochain element. */
    int i;

    SPTR buf_libre; /* Slot (dans producteur) a signaler lorsque
                     le tampon est libre. */

    /* On transmet au maitre les elements d'informations. */
    PUT_SYNC( TO_GLOBAL(&buf),          buf_ref,          fin_init );
    PUT_SYNC( TO_GLOBAL(&buf_libre),    buf_libre_ref,    fin_init );
    PUT_SYNC( TO_SPTR(RECEVOIR_ITEM),   buf_plein,       fin_init );
    PUT_SYNC( TO_SPTR(TERMINER),        terminer_consommateur, fin_init );

    /* On initialise le compteur vers l'element a recevoir. */
    c = 0;

    FIBER RECEVOIR_ITEM <* 1 *> {
        if(DEBUG) printf( "consommateur[RECEVOIR_ITEM]:c=%d\n", c );
        b[c] = buf;
        c += 1;
        SYNC( buf_libre );
    }

    FIBER TERMINER <* 1 *> {
        /* Avant de terminer, on imprime le tableau recu. */
        for( i = 0; i < c; i++ )
            printf( "b[%d]=%d\n", i, b[i] );
        SYNC( fin );
        TERMINATE;
    }
}
/*

```

28 Sep 00 14:57

producteur-consommateur.c

Page 3/3

```

*/
THREADED MAIN( int argc, char* argv[] )
{
    int n;                               /* Nombre d'elements du tableau. */
    int_GPTR buf_ref;                     /* Adresse du buf (dans le consommateur). */
    SPTR buf_plein;                       /* Slot du consommateur a signaler lorsque
                                          buf est plein. */
    SPTR buf_libre;                       /* Slot du producteur a signaler lorsque buf
                                          est libre. */
    SPTR_GPTR buf_plein_ref;              /* Reference dans le producteur a la slot
                                          du consommateur. */
    SPTR_GPTR buf_libre_ref;              /* Reference dans le consommateur a la slot
                                          du producteur. */
    SPTR terminer_consommateur;           /* Slot a signaler pour indiquer au
                                          consommateur que tout est fini. */

    /* On lit/definit la taille du tableau a transmettre.
       Valeur par default si argument absent == 10. */
    n = argc == 1 ? 10 : atoi(argv[1]);

    /* On invoque le producteur et le consommateur sur des processeurs.
       distincts. Il faut evidemment que ces processeurs soient disponibles. */
    assert( NUM_NODES >= 3 && "Ce programme doit etre execute sur au moins 3 processeurs." );

    if (DEBUG) printf( "Invocation du consommateur\n" );
    INVOKE( 1, consommateur,
            TO_GLOBAL(&buf_ref),
            TO_GLOBAL(&buf_plein),
            TO_GLOBAL(&buf_libre_ref),
            TO_GLOBAL(&terminer_consommateur),
            TO_SPTR(FIN_INIT_CONSOMMATEUR),
            TO_SPTR(CONSOMMATEUR_TERMINE) );

    FIBER FIN_INIT_CONSOMMATEUR <* 4 *> {
        if (DEBUG) printf( "Invocation du producteur\n" );
        INVOKE( 2, producteur,
                n,
                buf_ref,
                TO_GLOBAL(&buf_libre),
                TO_GLOBAL(&buf_plein_ref),
                TO_SPTR(FIN_INIT_PRODUCTEUR),
                TO_SPTR(PRODUCTEUR_TERMINE) );
    }

    FIBER FIN_INIT_PRODUCTEUR <* 2 *> {
        /* On a recu les sync slots a utiliser pour signaler si buf est
           vide/plein, ainsi que les references ou mettre ces sync slots:
           on les transmet au processus approprié. */
        PUT_SYNC( buf_plein, buf_plein_ref, FIN_INIT );
        PUT_SYNC( buf_libre, buf_libre_ref, FIN_INIT );
    }

    FIBER FIN_INIT <* 2 *> {
        /* Tout est en place: on active le producteur. */
        SYNC( buf_libre );
    }

    FIBER PRODUCTEUR_TERMINE <* 1 *> {
        /* Le producteur vient de se terminer: on signale le consommateur
           pour qu'il se termine a son tour. */
        SYNC( terminer_consommateur );
    }

    FIBER CONSOMMATEUR_TERMINE <* 1 *> {
        /* Le consommateur a lui aussi termine: on termine tout. */
        TERMINATE;
    }
}

```