(5)

Variante 2:

Moniteur Prod\_consom\_V2 {

I Buffer

I synchro

unsigned nbVides;

bool prochainType;

condition casePleine;

condition caseVide[2]; // 1 pour True 1 pour False

void deposer(Message mess, bool type) {

if(nbVides == 0 || prochainType != type) {

caseVide[type].wait();

// nbVides > 0 and prochainType == type

faireDepôt(mess, type);

prochainType = !prochainType;

nbVides-- ;

// réveiller un producteur du nouveau prochainType si le tampon n’est pas plein

if(nbVides != 0)

caseVides[prochainType].signal();

// Réveiller un consommateur

casePleine.signal();

}

void retirer(Message \*mess, bool \*type) {

if(nbVides == N)

casePleine.wait();

faireRetrait(mess, type);

nbVides++;

// Réveiller un producteur du prochainType

caseVide[prochainType].signal();

}

void initialiser() {

iVide = 0;

iPlein = 0;

nbVides = N;

prochainType = True;

}

}

Variante 3:

* Un producteur est bloqué dans déposer si le buffer est plein
* il est réveillé par un consommateur dans retirer quand il a fini de retirer un message
* Un consommateur de type T est bloqué dans retirer si le buffer est vide ou si le type T n’est pas en tête de buffer.
* Il est réveiller par un producteur de type T qui dépose en tête du buffer un message de type T (buffer etait vide)
* Il est réveillé par un consommateur qui finit de retirer si le buffer n’est toujours pas vide et il réveille un consommateur qui est de type en tête du buffer a present.

Moniteur Prod\_consom\_V3 {

// buffer

unsigned nbVides;

unsigned iPlein, iVide;

condition casePleine;

condition enTete[2]; // 1 pour True 1 pour False

void deposer(Message mess, bool type) {

if(nbVides == 0) {

caseVide.wait();

faireDepot(mess, type);

nbVides-- ;

// réveiller un consommateur de type type si le buffer était vide

if(nbVides == N - 1)

enTete[type].signal();

}

void retirer(Message \*mess, bool type) {

if(nbVides == N || buffer[iPlein].type != type)

enTete[type].wait();

faireRetrait(mess);

nbVides++;

if(nbVides != N)

enTete[buffer[iPlein].type].signal();

caseVide.signal();

}

void initialiser() {

iVide = 0;

iPlein = 0;

nbVides = N;

}

}

Processus Lecteur() {

while(true) {

demanderLecture();

// faire la lecture

terminerLecture();

}

}

Processus Redacteur() {

while(true) {

demanderEcriture();

// faire la lecture

terminerEcriture();

}

}