PC2R

TD1

Ex1. Processus

Q1

- OS:
- décide de l'attribution des ressources
- dirige le matériel par le logiciel
- régit les différents agents du système
- Scheduler
- partie de l'OS
- distribue l'exécution entre les processus
- stoppe l'exécution d'un processus, charge le contexte et lance l'exécution d'un autre processus
- Concurrence:
- exécution simultanée de plusieurs processus
- course aux ressources
- Préemption:
- interromp un processus en cours d'exécution et donne la main à un autre processus

Q2 Ordonnanceur: * Équitable * Vivacité/Progrès * Sur * Efficace

Q3

```
S  = [(x:= x+1;x;x:=x+1)||x:=2*x]
S' = [(x:=x+1;x:=x+1)||(wait(x=1);x:=2*x)]
```

- Séquence bloquante: g_1;g_2
- Séquence valeur 4: g_1;d_1;g_2;d_2
- Séquence valeur 3: g_1;d_1;d_2;g_2

Q4. Exemple d'ordonnancement P1 = a_1; a_2;...a_N, P2 = b_1; b_2;...b_N * Non-préemptif: une seule exécution possible * Préemptif: * a_1;a_2;b_1;b_2 * b_1;b_2;a_1;a_2 * a_1;b_1;a_2;b_2 * a_1;b_1;b_2;a_2 * b_1;a_1;a_2;b_2 * b_1;a_1;b_2;a_2 * En général: {M+N} choose {N} = {(M+N)!}/{N!+M!}

Q5. Etats de processus Cf. feuille, diag. à faire

Ex.2 Threads

Thread	**Processus**
Mémoire entièrement partagée	Propre mémoire
Meme PID	Propre PID
Environnement partagé	Propre environnement
CS peu couteux	CS couteux

Q1

Q2 Processus légers: pas de clonage, environnement partagé.

Q3. Gestion des threads La gestion des threads est faite par le processus qui les a créés (!= threads noyau géré par le système, ordonnancés *avec* les processus).

Q4. Contraintes d'ordonnancement

- non déterminisme
- difficulté de compréhension
- gestion de la mémoire partagée
- problèmes de sureté (*safety*), interblocage, verrou actif (*livelock*) - boucles non productives, lectures/écritures incohérentes
- vivacité (*liveness*)
- famine
- attente active

Ex3. Diner des philosophes

cf. feuille, diag à faire

```
P(g, d):
1    SLEEP
2    TAKE g     # Attente
3    TAKE d     # Attente
4    EAT
```

```
5    RELEASE d
6    RELEASE g
7    GOTO 1
```

État d'interlocage: tous les philosophes prennent g.

Q1 Tous les processus n'exécutent pas le meme programme selon leur parité.

Les processus pairs commencent par prendre à droite et les impairs à gauche ou inversement.

Q2. Chandy-Misra Flags sur les fourchettes

PC2R

TD 2

Ex1. Chemin de fer

Q1.

```
typedef enum { ROUGE = 0; VERT = 1 } Feu;
typedef enum { ALLUME = 1; ETEINT = 0} Detecteur;

Feu feu1 = ROUGE;
Feu feu2 = ROUGE;

Detecteur in1 = ETEINT;
Detecteur in2 = ETEINT;
Detecteur out1 = ETEINT;
Detecteur out2 = ETEINT;
```

Q2. 2^6 états possibles.

Q3.

- Incohérence: feu1 et feu2 verts, in1 et in2 allumés, out1 et out2 éteints
- Deadlock: feu1 et feu2 rouges, in1 et in2 allumés, out1 et out2 éteints

Q4. État avec in1 allumé ne passant pas jusqu'à un état avec out1 allumé.

Ex2. Rappels fair threads

Q1.

- **Fair Threads** Coopératifs (*A l'intérieur d'un scheduler!*)
- **Threads POSIX** Préemptifs

Q2.

```
#include "fthread.h"

void print1 (void *args) {
    while(1) {
        printf("Belle marquise ");
        ft_thread_cooperate();
    }
}

void print2 (void *args) {
    while(1) {
        printf("vos beaux yeux ");
        ft_thread_cooperate();
    }
}

void print3 (void *args) {
    while(1) {
        printf("me font mourir ");
        ft_thread_cooperate();
    }
}

void print4 (void *args) {
    while(1) {
        printf("d'amour\n");
        ft_thread_cooperate();
    }
}

int main () {
    ft_scheduler_t sched = ft_scheduler_create();

    ft_thread_create(sched, print1, NULL, NULL);
    ft_thread_create(sched, print2, NULL, NULL);
    ft_thread_create(sched, print3, NULL, NULL);
```

```
    ft_thread_create(sched, print4, NULL, NULL);

    ft_scheduler_start(sched);

    ft_exit();
    return 0;
}
```

Avec 4 schedulers: exécution comme sans les fair threads.

Ex3. Attentes actives

Q1.

```
int n = 0;
pthread_mutex_t mutex;

void *lecteur () {
    int my_n = 0;
    FILE * fic = fopen("/dev/urandom", "rb");
    for (;;) {
        pthread_mutex_lock(&fmutex);
        if (n != my_n) {
            int tmp, i;
            my_n = n;
            for (i = 0; i < n; i++) {
                fscanf(fic, "%d", &tmp);
                printf("%d", tmp);
            }
        }
        pthread_mutex_unlock(&fmutex);
    }
}

void *requete () {
    int my_n = 0;
    int tmp;
    printf("nombre");
    scanf(%d, &tmp);
    pthread_mutex_lock(&fmutex);
    n = tmp;
    pthread_mutex_unlock(&fmutex);
}

int main () {
```

```
    ...
    ...
}
```

Soucis: attente active dans le for du lecteur. Solution: variables de condition.

Q2.

```
pthread_cond_t condition =PTHREAD_COND_INITIALIZER;

void * lecteur () {
    ...
    pthread_mutex_lock(&fmutex);
    pthread_cond_wait(&condition);
    ...
}

void * requete () {
    ...
    pthread_mutex_unlock(&fmutex);
    pthread_cond_signal(&condition);
    ...
}
```

Ex4. Envoi/Attente

Q1.

```
ft_event_t evt;
ft_thread_await(evt);
printf("Evènement reçu\n");
ft_thread_cooperate ();
```

Q2.

```
ft_thread_cooperate_n(7);
...
```

PC2R

TD 3

Ex. 1 - Comptage au musée

```
let compteur = ref 0;
let cle = Mutex.create();

let rec entree nb =
    if nb > 0 then
        begin
            Mutex.lock cle
            compteur := !compteur + 1
            Mutex.unlock cle
            entrer (nb - 1)
        end

let sortie  =
    while true do
        if !compteur > 0 then
            begin
                Mutex.lock cle
                compteur := compteur - 1
                Mutex.unlock cle
            end
```

Ex. 2 - Scanner et imprimante

PC2R

TD 4

Ex1. Serveur d'echo

Q1. Serveur

```
public class Serveur extends Thread {
    BufferedReader inchan;
    DataOutputStream outchan;
    ServerSocket serv;
    Socket client;
```

```
    public Serveur() {
        try {ecoute = new ServerSocket(port);}
        catch (IOException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
        System.out.println("Serveur en écoute ")
    }

    public static void main (String [] args) {
        try {
            int port = Integer.parseInt(args[0]);
            serv = new ServerSocket(port);
            while (true) {
                client = serv.accept();
                try {
                    inchan = new BufferedReader(new InputStreamReader(client.getInputStream
                    outchan = new DataOutputStream(client.getOutputStream());
                    while (true) {
                        String command = inchan.readLine();
                        if (command.equals("")) {
                            System.out.println("Fin de connexion");
                            break;
                        }
                        outchan.write(command);
                    }
                } catch (IOException e) {
                    System.out.println("I/O error");
                    e.printStackTrace();
                }
                client.close();
            }
        }
    }
}
```

Q2. Pas de concurrence, un seul client à la fois. Les threads, c'est plus mieux.

Q3.

```
public class ServeurThread {
    ServerSocket serv;
    Socket client;
```

```
    public static void main (String [] args) {
        try {
            int port = Integer.parseInt(args[0]);
            serv = new ServerSocket(port);
            while (true) {
                client = serv.accept();
                Connection c = new Connection(client);
                c.start();
            }
        }
    } catch (IOException e) {
        System.out.println("I/O error");
        e.printStackTrace();
    }
}

public class Connexion extends Thread {
    private Socket client;
    private BufferedReader in;
    private DataOutputStream out;

    public Connexion(Socket client_socket) {
        client = client_socket;
        try {
            inchan = new BufferedReader(new InputStreamReader(client.getInputStream()));
            outchan = new DataOutputStream(client.getOutputStream());
        } catch (IOException e) {
            try { client.close(); } catch (IOException e1)
            {
                System.out.println("I/O error");
                e1.printStackTrace();
            }
            System.out.println("I/O error");
            e.printStackTrace();
        }
    }

    public void run () {
        while (true) {
            String command = inchan.readLine();
            if (command.equals("")) {
                System.out.println("Fin de connexion");
                break;
            }
            outchan.write(command);
        }
```

```
        client.close();
    }
}
```

Q6.

```
public class EchoServerPool{
    Vector<Connexion> clients;
    Vector<Socket>

    public static void main (String [] args) {

        int port = Integer.parseInt(args[0]);
        int capacity = Integer.parseInt(args[1]);
        EchoServer server = new EchoServer(port, capacity);
        server.start();
    }
}

public class EchoServer {

}

public class EchoClient {

}
```

Q8.

```
(* Compilation:     ocamlc -o server -thread -custom unix.cma threads.cma server.ml *)
let creer_serveur port max_con =
    let sock = Unix.socket Unix.PF_INET Unix.SOCK_STREAM 0
    and addr = Unix.inet_addr_of_string "127.0.0.1"
    in
        Unix.bind sock (Unix.ADDR_INET(addr, port))
        Unix.listen sock max_con;
        sock;;

let serveur_process sock service =
    while true do
        let (s, caller) = Unix.accept sock
        in
            ignore(Thread.create service (Unix.in_channel_of_descr s, Unix.out_channel_of_de
    done;;
```

```
let echo_service chans =
    let inchan = fst chans
    and outchan = snd chans
    in
        while true do
            let line = input_line inchan
            in
                output_string outchan (line ^ "\n");
                flush outchan
        done;;

let main () =
    let port = int_of_string Sys.argv.(1)
    and sock = creer_serveur port 4
    in
        serveur_process sock echo_service;;

let _ = main()
```

PC2R

TD 5

Rappels - canaux synchrones

```
new_channel
receive : 'a channel -> 'a event
send: 'a channel -> 'a -> unit event
sync: 'a event -> 'a
```

Ex1. Mobilité - Vente en ligne

Q1. Cf. feuille de TD.

Types des canaux: * n: string channel * c1, c2: string channel channel * i: (string, string channel channel) channel * s: (string, string channel) channel

Q2.

Vendeur (S)

```
let rec vendeur n =
    let (chan, prod) = sync (receive c_vendeur) in
        sync (send chan (prod ^ " " ^ (string_of_int n)));
    vendeur (n + 1)
```

Intermédiaire (I)

```
let chan_broker = new_channel ();;
let chan_seller = new_channel ();;

let rec intermediaire () =
    let (x, chan_buyer) = sync (receive chan_broker)
    and nu_c = new_channel () in
        sync(send chan_seller (nu_c, x));
        sync(send chan_buyer nu_c);
    intermediaire ()
```

Client (Cx)

```
let rec buyer args =
    let (a, n, c_buy, log, varlog) = args in
    if n == 0 then varlog := log else
    begin
        sync (send c_brok (a, c_buy))
        let chan = sync (receive c_buy) in
        let prod = sync (receive chan) in
        buyer (a, n-1, c_buy, log^prod^"\n", varlog)
    end
```

Main

```
let c_sell = new_channel ()
and c_brok = new_channel ()
and c_buy1 = new_channel ()
and c_buy2 = new_channel ()
and log1 = ref ""
and log2 = ref "";;

let main () =
    let _ = Thread.create seller 0 in
    let _ = Thread.create broker ()
    and t1 = Thread.create buyer ("thé", 3, c_buy1, log1)
    and t2 = Thread.create buyer ("café", 4, c_buy1, log1) in
    print_end_line !log1
    print_end_line !log2
```

Ex2.

```
let rec work (str, chan, n) =
    if n < max then
        Thread.delay (float_of_int (3+(Random_int 10))) /. 5.0
        let _ = sync (send chan str^" "^(string_of_int n)) in
        work str chan (n+1)
    else ()

let rec consumer () =
    let x = select [receive c_p, receive c_b, receive c_o] in
    print_end_line x;
    consumer ()
```

Ex6. Futurs

```
type 'a future = ('a channel * bool ref)

let spawn f arg =
    let c = new_channel ()
    and isdone = ref false in
    let run_future () =
        let res = f arg in
        isdone := true
        sync (send c res)
    in
    Thread.create run_future ();
    (c, isdone)

let isDone future =
    !(fst future)

let get future =
    sync (receive (snd future))
```