Laboratori 3 de SIOP
Ixent Cornella

1. Capitol

a) En el enunciat s'ens parla de TT i Capitol. Com ja sabem, el TT és el programa principal, i el capitol, en aquest lab, es representa com un fill del procés principal, un *fork*. Per saber que aquest està preparat, podem utilitzar els mètodes de comunicació entre processos familia, en aquest cas, podríem usar *kill*. La comanda *kill* envia al procés amb *pid* indicat la *signal* que escollim. Llavors, per comunicar Capitol amb TT, es podria reprogramar un *signal* (en aquest cas escollim el *signal SIGUSR1*), i que el capitol envïi un *kill* al pid del pare (amb *getppid()*), i mentres que el pare no rebï aquest senyal, es pausi.

b)
TT:
obrir_banc_temps()
comprovar_banc_temps()
pipe(per comunicar tt i capitol)
fork(capitol){
	tancar canals que substiturirem
	duplicar els canals necessaris
	//un cop fet aixo, ja esta preparat
	enviar_senyal_TT().
}
c)
```
int capitol = fork();
	if(capitol == 0){
	//redireccionar canal 0 i 1.
		dup2(fd, 0);   //fd = banc_temps
		dup2(p[1], 1);          //p = pipe
		close(fd);
		close(p[0]);
		close(p[1]);

		execl("./capitol", "./capitol", NULL);
	}
```
CODI CAPITOL:
```
kill(getppid(), SIGUSR1);   //enviar senyal

char c;
while (read(0, &c, 1) > 0) write(1, &c, 1);
exit(0);
```

2. TT

d) El TT (programa principal) s'encarrega de crear un procés fill capitol, que estarà comunicat amb el TT mitjançant una pipe. Abans de crear el capitol, el TT obrirà el banc del temps, que en aquest cas és un fitxer .dat. El capitol s'encarrega de tancar el seu canal estàndard de lectura i escritura, per llegir del banc del temps, i escriure en la pipe. Després, aquest envia una signal al TT indicant-li que ja està preparat. El TT llavors crea M replicants, que en aquest cas, també son processos fills. Aquests atacaran al banc del temps, amb funcions que ens proporcionen. Abans de res, els replicants redirigeixen el seu canal de lectura al de lectura de la pipe anteriorment creada, en la que escriu el Capitol. D'aquesta manera, el capitol escriu a la pipe, i els replicants ho llegeixen i converteixen les dades en temps amb les funcions donades. Finalment, aquests fan exit amb un codi particular, que després es va sumant al dels altres replicants per veure si el procés s'ha executat bé.

e)

```
TT:
crear_capitol //fet a l'anterior apartat
esperar_capitol;
for(M replicants){
        redirgeix_canals()
        inicialitza_contenidor
        while(llegeix_de_pipe()){
                calcula_temps;
        }
        dummy_exit()
}
suma_codis = suma_codis + codi //M vegades
```

f)

```
//CODI DEL CAPITOL EN L'ANTERIOR APARTATpause();
        close(fd); //no ens fan falta
        close(p[1]);

                for(int i = 0; i < M; i++){

                 int id = fork();
                 if(id == 0){
                        dup2(p[0], 0);
                        close(p[0]);
                        execl("./replicant", "./replicant", argv[2], NULL);
                 }
              }
          ...
```

```
            int st;
            while(waitpid(-1, &st, 0) > 0){
            if (WIFEXITED(st)) {
                        int es = WEXITSTATUS(st);
                        printf("Exit status was %d\n", es);
                        acum += es;

                    }
            }

            close(p[0]);
            if(dummy_testing(acum, ENIGMA2, TEAMNAME, BRONCEKEY) == 0) exit(0);
            else {
               printf("ERROR: NO s'ha pogut robar temps correctament. Missio fallida.\n");
               exit(-1);
            }
CODI REPLICANT:
        int N = atoi(argv[1]);
        char buffer[MAXTAM];

        if(N > 0 && N <= MAXTAM){
                int x;
                dummy_init(buffer, N);
                while((x = read(0, &buffer, N)) > 0)
                        dummy_calc(buffer, x);
                dummy_end();
                 }
      else {
         printf("ERROR: 'N' unitats de temps son massa grans per poder-les robar-les. Redueix
N.\n");
        exit(-1);
      }
```

3. Càlcul de valors òptims

El programa s'ha executat en una màquina virtual, fet que ralentitza considerablement l'execució. Tot i això, així queda la taula de temps en funció de M i N.

M = 1	M = 100	M = 5
N = 500.000	N = 500.000	N = 100.000
Temps = 8,5 minuts	Temps = 23 minuts	T = 11,25 minuts
M = 1	M = 25	M = 1000
N = 999.999	N = 100	N = 1
T = 9 minuts	T = +20 minuts	T = +30 minuts