Application temps réel

Marianne GRANDEMENGE
Médéric HURIER
Loïc MOUNIER

Sommaire

1. Introduction
2. Réflexion
3. Conception
4. Implémentation
5. Conclusion
6. Démonstration

Introduction

Présentation du problème

- Simulation logicielle d'un GPS
  - affichage en continu de la vitesse
  - avertissement en cas de dépassement
- Programmation de l'itinéraire et du comportement du conducteur
- Calcul de la vitesse à partir des données de géo-localisation

Aspect simulation

- Quelques limitations:
  - pas d'antenne GPS (input)
  - pas de périphérique de sortie (output)

⇨ Comment faire pour générer les localisations ?

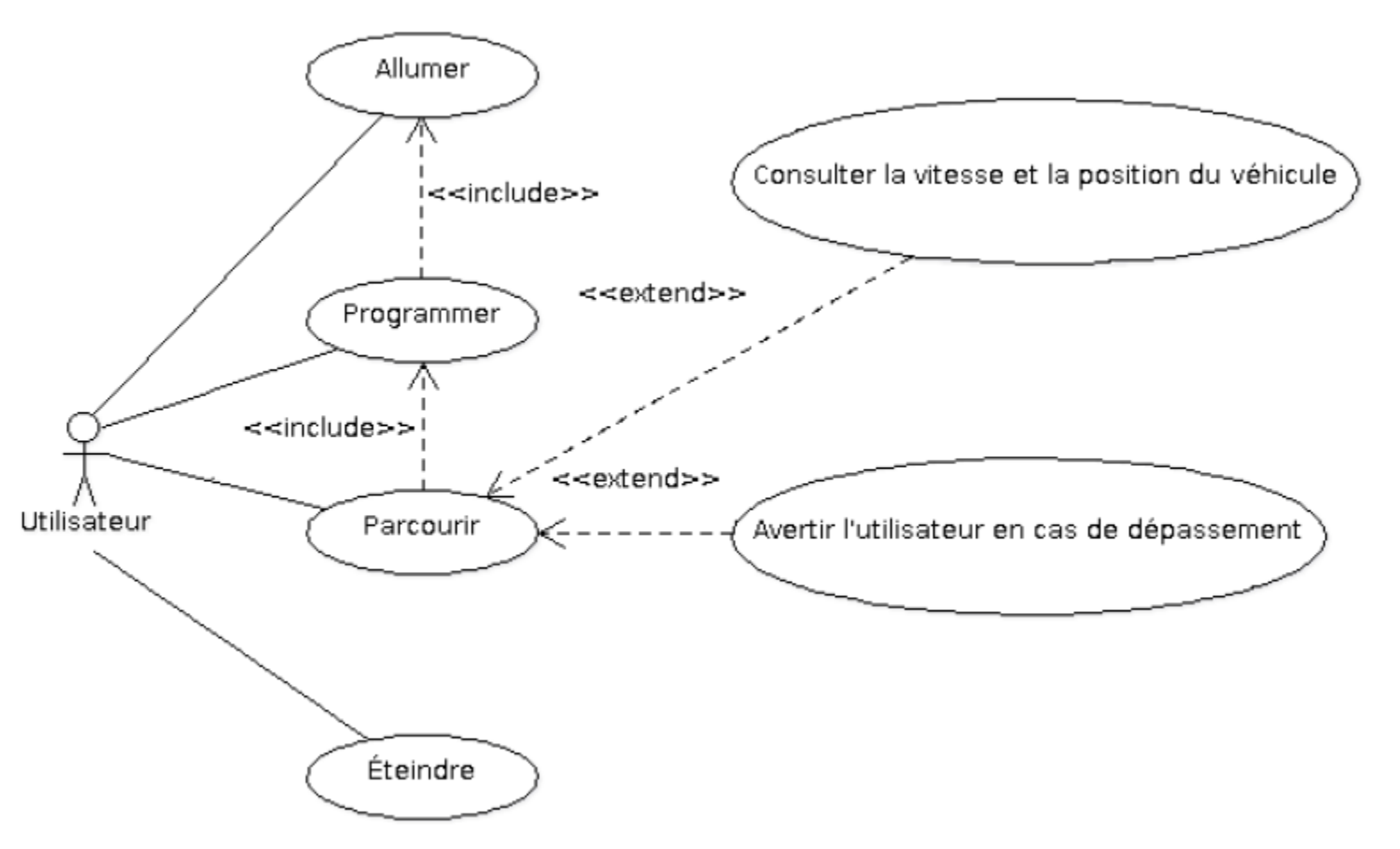
- Par simplification:
  - on considère que le véhicule se déplace sur une route assimilable à une droite
  - la route est découpée en tronçons

Enjeux

- Parallélisation des traitements
- Contrainte du temps réel
- Estimation de la position d'un véhicule

Réflexion

Fonctionnement du GPS



Géo-localisation

solution calcul

estimer la vitesse réelle à partir de la vitesse maximale du tronçon et du type de conduite

- dynamique
- aléatoire
- pas de données supplémentaires

solution jeu de données

utiliser un fichier de données pour fournir les positions du véhicule

- statique
- déterminé à l'avance
- 1 test = 1 nouveau fichier

Géo-localisation

solution calcul

estimer la vitesse réelle à partir de la vitesse maximale du tronçon et du type de conduite

solution jeu de données

utiliser un fichier de données pour fournir les positions du véhicule

- statique
- déterminé à l'avance
- 1 test = 1 nouveau fichier

dommage ...

Choix de l'IPC

Exigences:

- Rapide !
- Mémoire partagée
- Contrôle de l'exécution (synchronisation en fin de tâche)

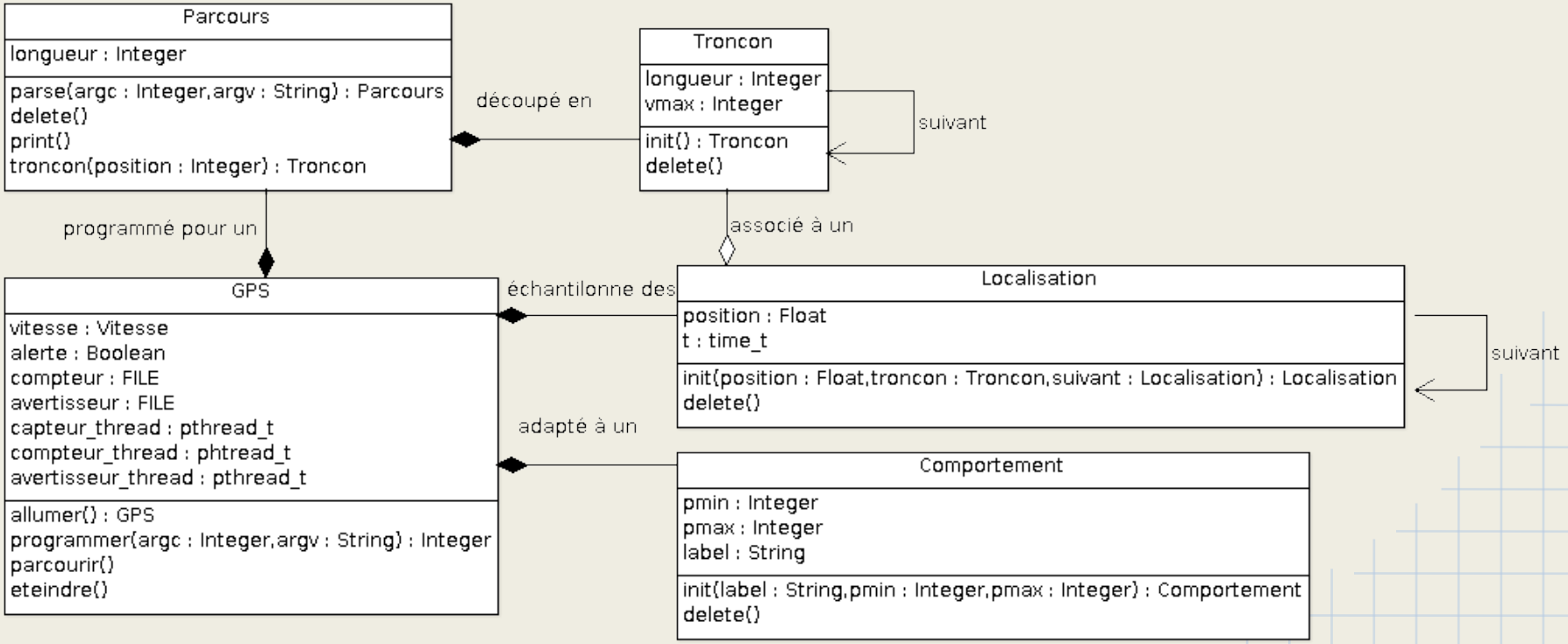
Notre solution : les Threads (processus légers)

D'après le cours de Mme Christine BOURJOT:

le changement de contexte lors d'une passage d'un processus à un autre est plus pénalisant au niveau performance que le passage d'un thread à un autre

Conception

Structures


Parcours
longueur : Integer
parse(argc : Integer,argv : String) : Parcours
delete()
print()
troncon(position : Integer) : Troncon
découpé en
Troncon
longueur : Integer
vmax : Integer
init() : Troncon
delete()
suivant
programmé pour un
associé à un
GPS
vitesse : Vitesse
alerte : Boolean
compteur : FILE
avertisseur : FILE
capteur_thread : pthread_t
compteur_thread : phtread_t
avertisseur_thread : pthread_t
allumer() : GPS
programmer(argc : Integer,argv : String) : Integer
parcourir()
eteindre()
échantilonne des
Localisation
position : Float
t : time_t
init(position : Float,troncon : Troncon,suivant : Localisation) : Localisation
delete()
suivant
adapté à un
Comportement
pmin : Integer
pmax : Integer
label : String
init(label : String,pmin : Integer,pmax : Integer) : Comportement
delete()

Exécutions parallèles

Traitements de chaque périphérique

- Capteur (antenne) : génère les données de localisation
- Compteur : affiche la vitesse du véhicule et sa position
- Avertisseur : affiche des alertes (visuels) en cas de dépassement

Problème de synchronisation:

- Pourquoi synchroniser ?
  - coûteux en temps (attente des autres tâches)
  - 1 tâche en écriture + 2 tâches en lecture non critique
- **= pas de synchronisation**
- Pour les threads, on peut utiliser les Mutex (similaire au Sémaphores)

Ressources critiques ?

Nous avons identifié:

- Vitesse du véhicule (capteur, compteur, avertisseur)
- Position du véhicule
  - vitesse maximale autorisée du tronçon (calcul du dépassement)

Les ressources sont partagées par les tâches, mais sont elles critiques ?

- délai d'affichage et d'avertissement le plus court possible
- jouer sur la fréquence d'échantillonnage / affichage

Pas de verrous sur les ressources

Algorithme : simulation

gps_simulation

Simule la position du véhicule en fonction de sa précédente vitesse

Paramètre(s) en entrée :

- gps

Algorithme :

```
Fonction gps_simulation (gps : GPS) : Localisation
Début
        vmax <- gps.localisation.troncon.vmax * 1000 / 3600.0
        vreel <- vmax
        pcent <- 0
        si gps.comportement.pmin ou gps.comportement.pmax alors
                pcent <- gps.comportement.pmin + rand() % (gps.comportement.pmax – gps.comportement.pmin)
                vreel <- vmax + vmax * pcent/100
        fsi

        dt <- maintenant - gps.localisations.t
        dp <- vreel * dt * acceleration
        dp_km = dp / 1000.0
        position <- gps.localisations.position + dp_km

        localisation <- localisation_init(position, parcours_troncon(gps.parcours, position), gps.localisations)
        localisation.t <- maintenant

        retourne localisation
Fin
```

Algorithme : capteur

gps_capteur

Fonction asynchrone échantillonnant les relevés (simulés) du satelite

Paramètre(s) en entrée :

- gps

Algorithme :

```
Fonction gps_capteur (gps : GPS)
Début
        tant que continuer faire
            Loc <- gps_simulation(gps)
              Dp <- loc.position – gps.localisations.position
              Dt <- (loc.t  - gps.localisations.t) * acceleration
            Gps.vitesse <- (dp/dt) * 3600
            si loc.position >= gps.parcours.longueur alors
                        continuer = 0
            fsi

            gps.localisations <- loc
            attendre 1 seconde
      ftant
Fin
```

Algorithme : compteur

gps_compteur

Fonction asynchrone qui permet d'afficher la valeur du compteur.

Paramètre(s) en entrée :

- Gps

Algorithme :

```
Fonction gps_compteur(gps : GPS)
Début
        tant que continuer faire
                afficher(gps.vitesse, gps.troncon.position)
                 attendre 1 seconde
        ftant
      horodatage(gps.compteur)
Fin
```

Algorithme : avertisseur

gps_avertisseur

Fonction asynchrone avertissant le conducteur en cas de dépassement de vitesse. On avertit une seule fois l'utilisateur jusqu'à ce qu'il redescende sous la vitesse maximum autorisée

Paramètre(s) en entrée :

- Gps

Algorithme :

```
Fonction gps_avertisseur(gps : GPS)
Début
        tant que continuer faire
                depassement <- gps.vitesse – gps.localisations.troncon.vmax
                si !gps et depassement > 0 alors
                        horodatage(gps.avertisseur, « alerte »)
                        gps.alerte <- 1
                sinon
                si gps.alerte et depassement <= 0 alors
                                        horodatage(gps.avertisseur, « . »)
                                        gps.alerte <- 0
                        fsi
                fsi
                attendre 1 seconde

        ftant
    horodatage(gps.avertisseur, « FIN » )
Fin
```

Implémentation

Environnement de dév.

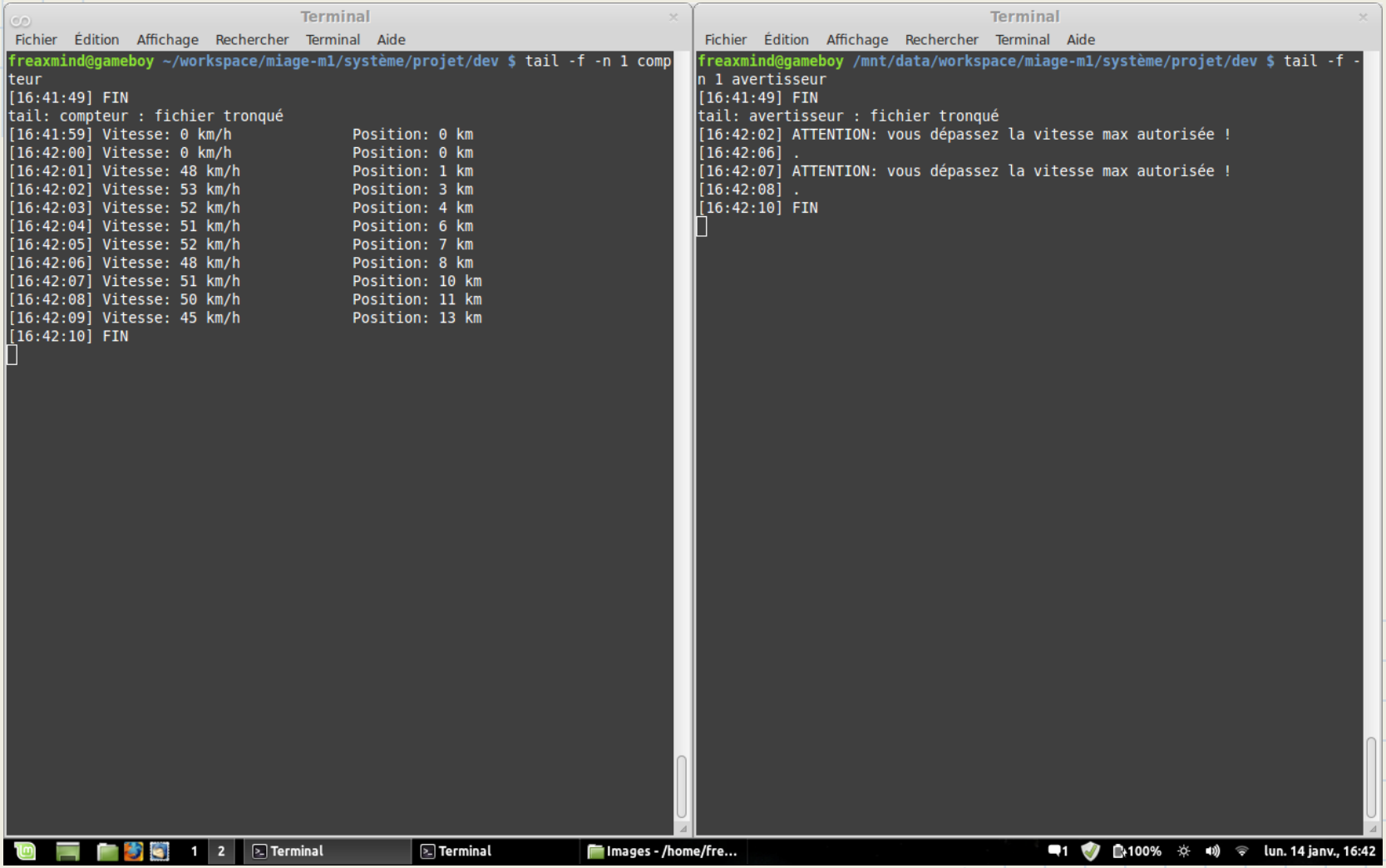
Que Vive l'Open Source !

Plateforme:

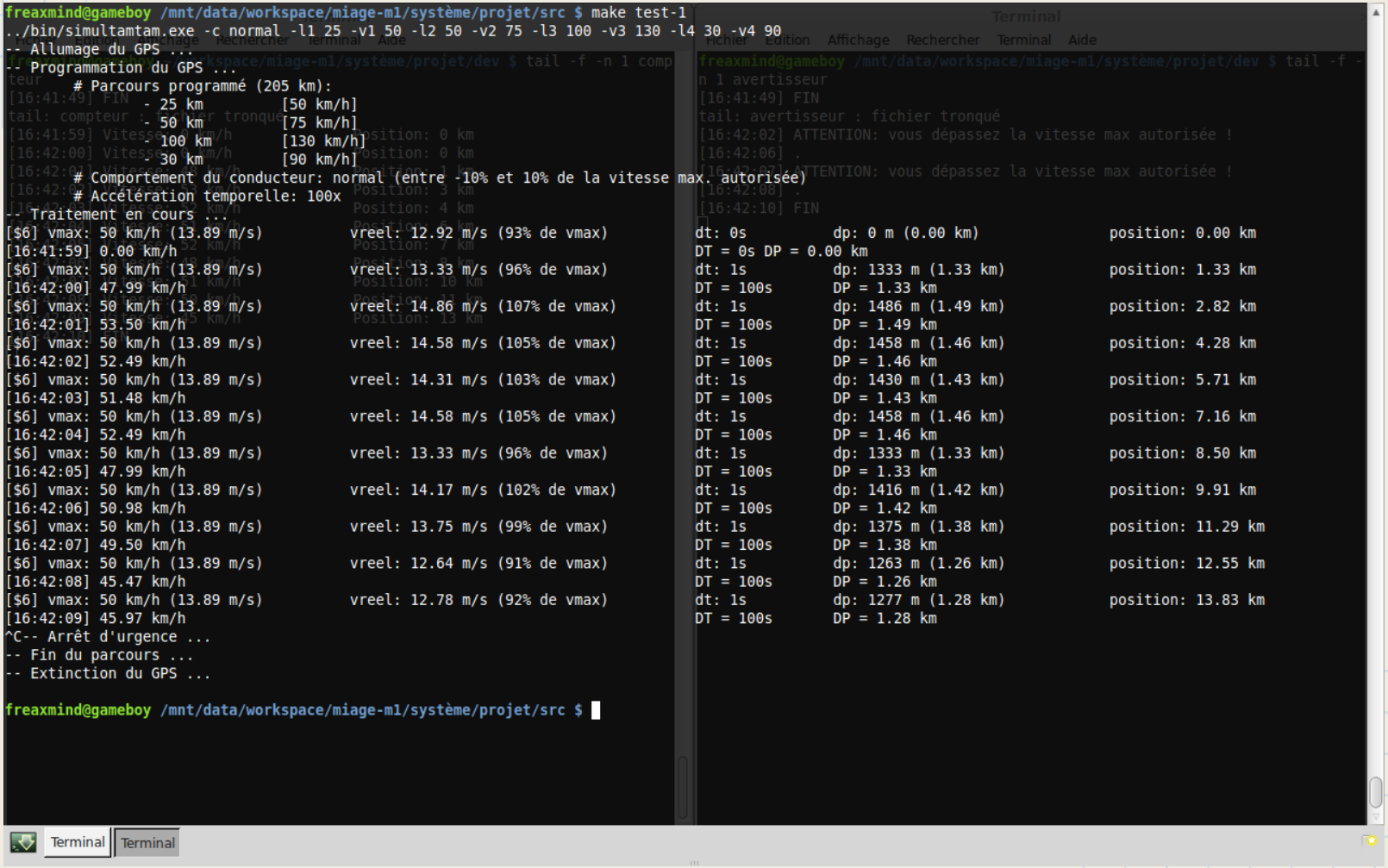
- Linux Mint 14 (dérivé de Ubuntu)
- Langage C
- GCC 4.7.2
- Make 3.81
- Thread POSIX (pthread)
  - gcc -Wall -ansi -g *.o -o ../bin/simultamtam.exe -lpthread

Pas de bibliothèques supplémentaires ou non natives !

Interface


Terminal
Fichier Édition Affichage Rechercher Terminal Aide
freaxmind@gameboy ~/workspace/miage-m1/système/projet/dev $ tail -f -n 1 compteur
[16:41:49] FIN
tail: compteur : fichier tronqué
[16:41:59] Vitesse: 0 km/h Position: 0 km
[16:42:00] Vitesse: 0 km/h Position: 0 km
[16:42:01] Vitesse: 48 km/h Position: 1 km
[16:42:02] Vitesse: 53 km/h Position: 3 km
[16:42:03] Vitesse: 52 km/h Position: 4 km
[16:42:04] Vitesse: 51 km/h Position: 6 km
[16:42:05] Vitesse: 52 km/h Position: 7 km
[16:42:06] Vitesse: 48 km/h Position: 8 km
[16:42:07] Vitesse: 51 km/h Position: 10 km
[16:42:08] Vitesse: 50 km/h Position: 11 km
[16:42:09] Vitesse: 45 km/h Position: 13 km
[16:42:10] FIN
Terminal
Fichier Édition Affichage Rechercher Terminal Aide
freaxmind@gameboy /mnt/data/workspace/miage-m1/système/projet/dev $ tail -f -n 1 avertisseur
[16:41:49] FIN
tail: avertisseur : fichier tronqué
[16:42:02] ATTENTION: vous dépassez la vitesse max autorisée !
[16:42:06] .
[16:42:07] ATTENTION: vous dépassez la vitesse max autorisée !
[16:42:08] .
[16:42:10] FIN
1 2 Terminal Terminal Images - /home/fre... 100% lun. 14 janv., 16:42

Interface (debug)



Erreurs et exceptions

Implémenté:

- Nombre d'argument (manque ou incorrect)
- Vitesse et longueur entier positif
- Comportement inconnu

Manquant:

- Nom des vitesses et longueurs (v1, l1 ...)
- Rang des arguments

Conclusion

Bilan

- Le programme fonctionne
- Le code est propre
- La solution est simple, rapide et évolutive

Commentaires personnels

Sur le projet:

- pas de vrai capteur
- pas de comparatif entre processus / threads
- **"simple" mais intéressant**

Sur le module:

- structures et méthodes utiles !
- pas assez de TP machine

Démonstration

1. Exemple d'utilisation
2. Cas d'erreurs
3. Cas nominal
4. Cas normaux