TP Architecture d'un système d'exploitation
Diagramme d'implantation des traitements
(pour deux processus)

29th January 2004

1 Processus ComUser

1.1 Primitives

1. Nom: RecevoirRequete
   Description: Lit sur input le nom du fichier à exécuter et le nom du fichier de données.
   Nom d'implantation: recReq
   Paramètres entrants: aucun
   Paramètres sortants: req (Requete)
   Préconditions: aucune
   Postconditions: La fonction renvoie, dans son paramètre sortant "req", le nom des deux fichiers, qui sont lus sur l'input standard.
   Codes d'erreur: ERR_LIRE_INPUT, ERR_ALLOC_MEMOIRE

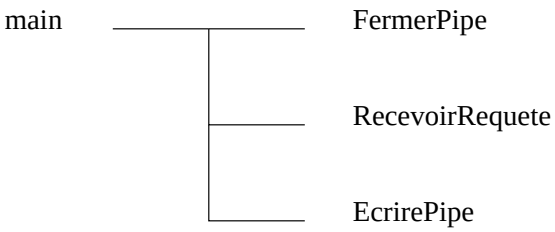
2. Nom: EcrirePipe
   Description: Ecrit de l'information sur un pipe anonyme.
   Nom d'implantation: ecrirePipe
   Paramètres entrants: fd (int), buf (void *), taille (int)
   Paramètres sortants: nbr de bytes lus (int)
   Préconditions: le fd existe et pointe bien vers un pipe ouvert en écriture
   Postconditions: Les "taille" premiers bytes situés en mémoire à partir du l'adresse "buf" ont bien été écrits sur le pipe décrit par le file descriptor "fd".
   Codes d'erreur: ERR_ECRIRE_PIPE.

3. Nom: FermerPipe
   Description: Ferme l'une des deux directions de la communication.
   Nom d'implantation: fermerPipe
   Paramètres entrants: fd (int)

Paramètres sortants: aucun
Préconditions: le fd est valide
Postconditions: l'accès au pipe par le file descriptor donné en input est bien fermé.
Codes d'erreur: ERR_FERMER_PIPE.

4. Nom: main
   Description: Appelle "fermerPipe", puis "recevoirRequete", fait des vérifications (fichier existe, etc...), appelle "ecrirePipe".
   Nom d'implantation: main
   Paramètres entrants: aucun
   Paramètres sortants: aucun
   Préconditions: aucune
   Postconditions: aucune
   Codes d'erreur: ERR_FERMER_PIPE, ERR_ECRIRE_PIPE, ERR_LIRE_INPUT, ERR_ALLOC_MEMOIRE.

1.2 Arbre d'appel



2 Processus Router

2.1 Primitives

1. Nom: main
   Description: Appelle "Init", puis "AiguillerRequete".
   Nom d'implantation: main
   Paramètres entrants: fichier_config (char *)
   Paramètres sortants: aucun
   Préconditions: aucune
   Postconditions: aucune
   Codes d'erreur: ERR_CREER_PIPE, ERR_FERMER_PIPE, ERR_INIT_MEM_PART, ERR_ATTACH_MEM_PART, ERR_INIT_CHARGES, ERR_CREER_SOCKET_STR, ERR_FORK, ERR_EXEC, ERR_SELECT, ERR_OUVRIR_FICHIER, ERR_LIRE_FICHIER, ERR_FERMER_FICHIER.

2. Nom: Init
   Description: Appelle "CreerPipe" pour créer deux pipes anonymes (l'un pour communiquer avec le processus ComUser, l'autre pour communiquer

avec le processus GestRequete), crée et initialise une mémoire partagée en appelant "InitMemPartagee", puis y initialise les charges et les identités de chacun de ses voisins et de lui-même, en appelant "InitTableCharges". Crée les sockets en mode stream pour chacun de ses voisins (en appelant "CreerSocketStream"). Créer ses deux processus fils, ComUser et GestReq, puis appelle "FermerPipe" pour fermer l'un des sens de la communication.
Nom d'implantation: init
Paramètres entrants: fichier_config (char *)
Paramètres sortants: aucun
Préconditions: aucune
Postconditions: Les diverses initialisations évoquées dans la description ci-dessus sont effectuées correctement.
Codes d'erreur: ERR_CREER_PIPE, ERR_FERMER_PIPE, ERR_INIT_MEM_PART, ERR_ATTACH_MEM_PART, ERR_INIT_TABLE_CHARGES, ERR_CREER_SOCKET_STR, ERR_FORK, ERR_EXEC, ERR_OUVRIR_FICHIER, ERR_LIRE_FICHIER, ERR_FERMER_FICHIER .

3. Nom: AiguillerRequete
   Description: Fait un appel système "select" pour être réveillé quand quelque chose (i.e. une requete) est présent sur le pipe ou sur le socket. Lit la requête, sur le pipe ou le socket selon le cas (appel de "LireReqPipe" ou de "LireReqSocket"). Apelle ensuite "ChercherChargeMin", et selon le résultat (soit nou sommes la machine la moins chargée soit pas) on envoie la requête au processus GestRequete (par appel de "EcrireReqPipe"), ou à la machine voisine de plus petite charge (par appel de "EcrireReqSocket").
   Nom d'implantation: aiguillerReq
   Paramètres entrants: aucun
   Paramètres sortants: resultat (int)
   Préconditions: L'appel à Init s'est déroulé sans incident.
   Postconditions: le requête a été correctement traitée (envoyée sur pipe ou sur socket). Codes d'erreur: ERR_SELECT, ERR_LIRE_SOCK_STREAM, ERR_ECRIRE_SOCK_STREAM, ERR_OUVRIR_FICHIER, ERR_LIRE_FICHIER, ERR_ECRIRE_FICHIER, ERR_FERMER_FICHIER, ERR_ECRIRE_PIPE, ERR_LIRE_PIPE, ERR_LIRE_REQ_PIPE, ERR_LIRE_REQ_SOCK, ERR_ECRIRE_REQ_PIPE, ERR_ECRIRE_REQ_SOCK (faut-il garder ces 4 derniers codes d'erreur ? Voir rmqs dans le dico des erreurs).

4. Nom: ChercherChargeMin
   Description: Consulte la mémoire partagée pour chercher la machine qui a la charge minimale.
   Nom d'implantation: chargeMin
   Paramètres entrants: aucun
   Paramètres sortants: machine_min char[4]
   Préconditions: Table correctement initialisée
   Postconditions: Renvoie l'addresse IP de la machine qui, dans la table des

charges, a l'addresse minimale.
Codes d'erreur: aucun

5. Nom: LireReqPipe
   Description: Lire, sur un pipe anonyme, le nom du fichier exécutable et du fichier contenant les données d'input. Utilise la primitive "LirePipe".
   Nom d'implantation: lireReqPipe
   Paramètres entrants: fd (int)
   Paramètres sortants: req (Requete)
   Préconditions: Le paramètre "fd" pointe bien vers un pipe anonyme correctement initialisé.
   Postconditions: Le paramètre sortant "req" contient bien les deux noms de fichier lus sur le pipe.
   Codes d'erreur: ERR_LIRE_PIPE.

6. Nom: EcrireReqPipe
   Description: Ecrire, sur un pipe anonyme, le nom du fichier exécutable et du fichier contenant les données d'input. Utilise la primitive "EcrirePipe".
   Nom d'implantation: ecrireReqPipe
   Paramètres entrants: fd (int), req (Requete)
   Paramètres sortants: aucun
   Préconditions: Le paramètre "fd" pointe bien vers un pipe anonyme correctement initialisé.
   Postconditions: Les deux strings présents dans le paramètre "req" sont correctement écrits sur le pipe référencé par le descripteur "fd".
   Codes d'erreur: ERR_ECRIRE_PIPE.

7. Nom: LireReqSocket
   Description: Lire, sur un socket, un exécutable et ses données d'input, et les écrire sur deux fichiers séparés. Utilise les primitives "LireSocketStream" et "EcrireFichier".
   Nom d'implantation: lireReqSock
   Paramètres entrants: sd (int)
   Paramètres sortants: req (Requete)
   Préconditions: Le paramètre "sd" pointe bien vers un socket stream correctement initialisé.
   Postconditions: Les deux fichiers sont correctement écrits sur le disque et leur nom est correctement renvoyé par la primitive dans le paramètre sortant "req".
   Codes d'erreur: ERR_OUVRIR_FICHIER, ERR_FERMER_FICHIER, ERR_LIRE_SOCK_STREAM, ERR_ECRIRE_FICHIER.

8. Nom: EcrireReqSocket
   Description: Lire le fichier exécutable et le fichier des données sur disque, et les écrire sur un socket. Utilise les primitives "EcrireSocketStream" et "LireFichier".

Nom d'implantation: ecrireReqSock
Paramètres entrants: req (Requete), sd (int)
Paramètres sortants: aucun
Préconditions: Le paramètre "req" est bien intialisé avec deux noms de fichiers existants, le paramètre "sd" pointe bien vers un socket stream correctement initialisé.
Postconditions: Les deux fichiers sont correctement écrits sur le socket.
Codes d'erreur: ERR_OUVRIR_FICHIER, ERR_FERMER_FICHIER, ERR_ECRIRE_SOCK_STREAM, ERR_LIRE_FICHIER.

9. Nom: CreerPipe
   Description: Crée un pipe anonyme.
   Nom d'implantation: CreerPipe
   Paramètres entrants: aucun
   Paramètres sortants: fd[2] (int *)
   Préconditions: aucune
   Postconditions: Le pipe a été correctement créé
   Codes d'erreur: ERR_CREER_PIPE

10. Nom: LirePipe
    Description: Lit de l'information sur un pipe anonyme.
    Nom d'implantation: lirePipe
    Paramètres entrants: fd (int), taille (int)
    Paramètres sortants: buf(void *)
    Préconditions: le fd existe et pointe bien vers un pipe ouvert
    Postconditions: Les "taille" premiers bytes lus sur le pipe pointé par file descriptor "fd" sont copiés en mémoire à partir du l'adresse "buf" .
    Codes d'erreur: ERR_LIRE_PIPE

11. Nom: EcrirePipe
    Description: Ecrit de l'information sur un pipe anonyme.
    Nom d'implantation: ecrirePipe
    Paramètres entrants: fd (int), buf (void *), taille (int)
    Paramètres sortants: aucun
    Préconditions: le fd existe et pointe bien vers un pipe ouvert
    Postconditions: Les "taille" premiers bytes situés en mémoire à partir du l'adresse "buf" ont bien été écrits sur le pipe décrit par le file descriptor "fd".
    Codes d'erreur: ERR_ECRIRE_PIPE.

12. Nom: FermerPipe
    Description: Ferme l'une des deux directions de la communication.
    Nom d'implantation: fermerPipe
    Paramètres entrants: fd (int)
    Paramètres sortants: aucun
    Préconditions: le fd existe et pointe bien vers un pipe
    Postconditions: l'accès au pipe par le file descriptor donné en input est

bien fermé.
Codes d'erreur: ERR_FERMER_PIPE

13. Nom: CreerSocketStream
    Description: Crée des socket stream pour tous les voisins de la machine courante.
    Nom d'implantation: creerSockStream
    Paramètres entrants: fichier_config (char *)
    Paramètres sortants: aucun
    Préconditions: le fichier de config contenant l'adresse de tous les voisins est correctement formatté
    Postconditions: une connexion stream existe entre notre machine courante et toutes les machines voisines
    Codes d'erreur: ERR_CREER_SOCK_STREAM

14. Nom: LireSocketStream
    Description: Lit de l'information sur un socket stream.
    Nom d'implantation: lireSockStream
    Paramètres entrants: sd (int), buf (void *), taille (int)
    Paramètres sortants: buf (void *
    Préconditions: le sd existe et pointe bien vers un socket correctement initialisé.
    Postconditions: Les "taille" premiers bytes lus sur le socket pointé par socket descriptor "sd" sont copiés en mémoire à partir du l'adresse "buf" .
    Codes d'erreur: ERR_LIRE_SOCK_STREAM

15. Nom:EcrireSocketStream
    Description: Ecrit de l'information sur un socket stream.
    Nom d'implantation: ecrireSockStream
    Paramètres entrants: sd (int), buf (void *), taille (int)
    Paramètres sortants: aucun
    Préconditions: le sd existe et pointe bien vers un socket correctement initialisé.
    Postconditions: Les "taille" premiers bytes situés en mémoire à partir du l'adresse "buf" ont bien été écrits sur le socket décrit par le socket descriptor "sd".
    Codes d'erreur: ERR_ECRIRE_SOCK_STREAM

16. Nom: OuvrirFichier
    Description: Ouvre un fichier, soit en mode "écriture+création", soit en mode "lecture". Le mode choisi est spécifié par le booléen passé en paramètre.
    Nom d'implantation: ouvrirFichier
    Paramètres entrants: nom (string), mode (booléen)
    Paramètres sortants: fd (int)
    Préconditions: En cas de demande d'ouverture en lecture, le nom du fichier désigne un fichier existant.
    Postconditions: Le fichier spécifié est ouvert soit en mode lecture soit est

créé et ouvert en écriture, selon le booléen passé en paramètre.
Codes d'erreur: ERR_OUVRIR_FICHIER .

17. Nom: LireFichier
    Description: Lit de l'information sur un fichier.
    Nom d'implantation: lireFichier
    Paramètres entrants: fd (int), taille (int)
    Paramètres sortants: buf (void *)
    Préconditions: le fd existe et pointe bien vers un fichier ouvert en lecture
    Postconditions: Les "taille" premiers bytes lus sur le fichier pointé par file descriptor "fd" sont copiés en mémoire à partir du l'adresse "buf" .
    Codes d'erreur: ERR_LIRE_FICHIER

18. Nom: EcrireFichier
    Description: Ecrit de l'information sur un fichier.
    Nom d'implantation: ecrireFichier
    Paramètres entrants: fd (int), taille (int)
    Paramètres sortants: buf (void *)
    Préconditions: le fd existe et pointe bien vers un fichier ouvert en écriture
    Postconditions: Les "taille" premiers bytes situés en mémoire à partir du l'adresse "buf" ont bien été écrits sur le pipe décrit par le file descriptor "fd".
    Codes d'erreur: ERR_ECRIRE_FICHIER

19. Nom: FermerFichier
    Description: Ferme un fichier.
    Nom d'implantation: fermerFichier
    Paramètres entrants: fd (int)
    Paramètres sortants: aucun
    Préconditions: Le fd pointe vers un fichier existant.
    Postconditions: Le fichier est bien fermé.
    Codes d'erreur: ERR_FERMER_FICHIER .

20. Nom: InitMemPartagée
    Description: Crée et attache une mémoire partagée.
    Nom d'implantation: initMemPart
    Paramètres entrants: aucun
    Paramètres sortants: adresse (void *)
    Préconditions: aucune
    Postconditions: mémoire partagée correctement attachée à l'espace d'adressage du processus
    Codes d'erreur: ERR_INIT_MEM_PART, ERR_ATTACH_MEM_PART.

21. Nom: InitTableCharges
    Description: Lit les fichiers de configuration, et initialise les charges et les identificateurs dans la table des charges qui se trouve en mémoire partagée.
    Nom d'implantation: InitTableCharges
    Paramètres entrants: fichier_config (char *)

Paramètres sortants: aucun
Préconditions: Les fichiers de configuration sont corrects et la mémoire partagée a été correctement initialisée.
Postconditions: La mémoire partagée contient les addresses IP correctes de tous les voisins, et initialise les charges à 0
Codes d'erreur: ERR_INIT_TABLE_CHARGES, ERR_OUVRIR_FICHIER, ERR_LIRE_FICHIER, ERR_FERMER_FICHIER

2.2 Arbre d'appel