Programme principal et arguments

->Programme système : Langage C

->Fonction principal main()

int main(int argc, char\*argv[])

argc : contient le nombre d’argument passé en parametre par l’interpreteur de commande au programme appelé

argv : contient les agurments au format texte.

Exemple

$./prog arg1 428 toto

argc=4

argv[0]= « ./prog»

argv[1]= « arg1»

argv[2]= « 428»

argv[3]= « toto»

Schéma mémoire d’un processus

La mémoire d’un processus est constituée :

-d’un segment texte : ce sont les instructions exécutées par le processus. Le segment est généralement en lecture seule et peut être partagé

-Segment de données (int max =99 ;)

-Segment de données non initialisées (int tab[100]) : les blocs sont 0 ou NULL

-Pile : sert à la sauvegarde d’informations, en particulier lors de l’appel de fonction, au retour et au stockage de données

-Tas : sert à l’allocation dynamique

|  |
| --- |
| Argument et Variable |
| Pile |
| Tas |
| Données non initialisées |
| Données initialisées |
| Texte |

La commande size fournit la taille des différents segments mémoire

$ size /bin/ts

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Text | Data | Bss | Others | Dec | Hex |
| 24576 | 4096 | 0 | 7884 | 36556 | 8ccc |

Primitives memoires

-void\* malloc(size\_t S)

Alloue une zone mémoire dans le tas de la taille spécifiée l’adresse défault est retournée le contenu est arbitraire

Variable d’environnement

-

Les limitations ressources

-int getrlimit(int ressource , struct rlimit \*rlptr)

-int setrlimit(int ressource , struct rlimit \*rlptr)

-struct rlimit{

rlimit rlim\_cur ; /\*limite courante\*/

rlimit rlim\_max ;/\*limite max\*/

Ressources

RLIMIT\_CORE= taille max du fichier code généré en cas d’erreur fatal

RLIMIT\_CPU=maximum de temps CPU en secondes

RLIMIT\_DATA= taille max du segment de données (initialisé ou non)

RLIMIT\_FSIZE=taille max des fichiers créés

RLIMIT\_NOFILE= Nombre maximal de fichier ouvert

= nombre max de processus enfants

RLIMIT\_STACK = Taille max de la pile

Exercice 4 : Afficher le nombre max de fichier ouvert pour le processus courant etfixer le nombre max d’enfants a 2

Identifier un processus

pid\_t getpid() ; id processus

pid\_t getppid() ; id processus père

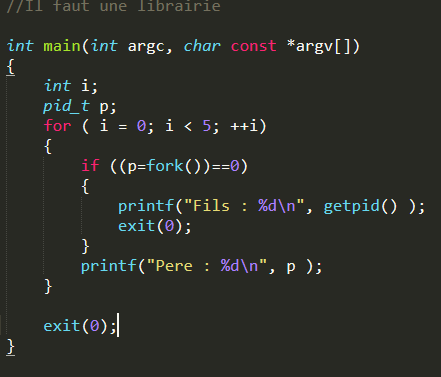
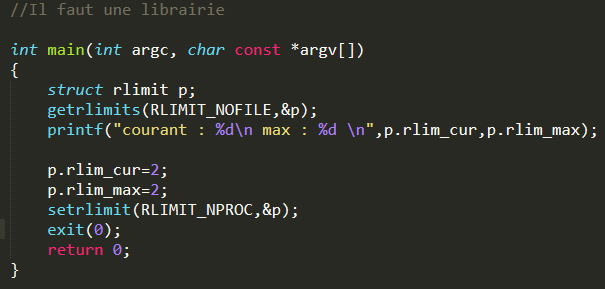
uid\_t getuid() ; id de l’utilisateur

uid\_t geteuid() ; id de l’utilisateur effectif

gid\_t getgid(); id du groupe

gid\_t getegid();id du groupe effectif

Exo5 : Créer 5 processus , affiche le pid du processus créer et les processus fils affiche leurs pid



Terminaison de processus

-void exit (int status)

Effectue un « ménage » et quitte le processus avec le statu status

-void \_exit (int status)

Quitte le processus avec un statu status

-int atexit(void(\*func)(void))

Spécifie les opérations a réaliser lors de la terminaison de processus

-pid\_t wait(int \*statloc)

Attend la fin d’un processus fils et recuperer ses valeurs de retour (dans statloc)

Retourne le pid du fils terminé

-pid\_t waitpid (pid\_t pid ,int\* statloc, int options)

Attends la fin du processus fils pid et récupère sa valeur de retour dans statloc

Les options servent éventuellement à avoir un appel non bloquant

Exo6 : Ecrire un programme qui créé 10 processus fils qui affichent leurs identifiants. Le processus père affichera les ids des processus fils dans l’ordre qu’ils se terminent

