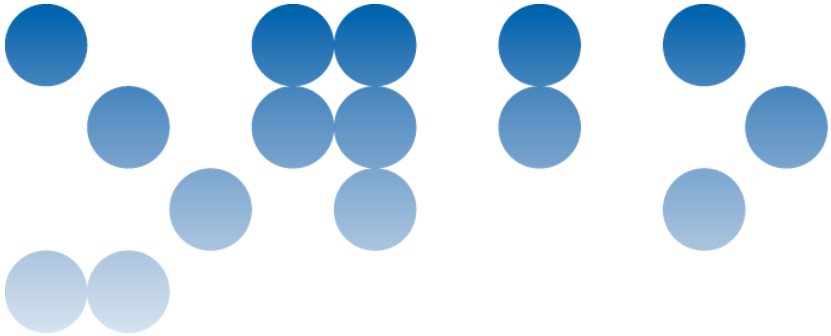

PRA2
Disseny i implementació d'aplicacions multiprocés en Linux

Sistemes Operatius

Programa

2018-02

Estudis d'Informàtica, Multimèdia i Telecomunicació

EIMT.UOC.EDU

UOC
Universitat Oberta de Catalunya

Presentació

Aquesta pràctica planteja un seguit d'activitats amb l'objectiu que l'estudiant pugui aplicar sobre un sistema Unix alguns dels conceptes introduïts als darrers mòduls de l'assignatura.

L'estudiant haurà de realitzar un seguit d'experiments i respondre les preguntes plantejades i escriure uns petits programes en llenguatge C.

Competències

Transversals:

- Capacitat per a adaptar-se a les tecnologies i als futurs entorns actualitzant les competències professionals

Especifiques:

- Capacitat per a analitzar un problema en el nivell d'abstracció adequat a cada situació i aplicar les habilitats i coneixements adquirits per a abordar-lo i resoldre'l
- Capacitat per a dissenyar i construir aplicacions informàtiques mitjançant tècniques de desenvolupament, integració i reutilització

Competències pròpies:

- L'ús i aplicació de les TIC en l'àmbit acadèmic i professional

Objectius

Aquesta pràctica té com a objectiu que l'alumne apliqui els conceptes introduïts als 3 darrers mòduls de l'assignatura sobre un sistema Unix i que s'introdueixi de forma progressiva en els aspectes avançats de la programació, depuració i execució d'aplicacions en els sistemes Unix.

Restriccions en l'ús de funcions de C

Per realitzar la pràctica heu d'utilitzar les crides a sistema d'Unix. **No es podran utilitzar funcions de C d'entrada/sortida** (getc, scanf, printf,...), en el seu lloc s'utilitzaran les crides a sistema read i write. Sí que podeu utilitzar funcions de C per al formatatge de cadenes (sprintf, sscanf, ...).

Enunciat

A aquesta pràctica haureu de realitzar un seguit d'experiments, desenvolupar petits programes en C i respondre justificadament les preguntes plantejades. Podeu fer els experiments/programes sobre qualsevol sistema Unix al que tingueu accés.

Us facilitem el fitxer pr2so.zip amb una sèrie de fitxers que us seran útils per fer i validar la pràctica. Descompacteu-lo executant la comanda unzip pr2so.zip.

Com a referència, a cada pregunta us aconsellem una possible temporització per tal de poder acabar la practica abans de la data límit. També s'indica el pes de cada pregunta a l'avaluació final de la pràctica.

El pes d'aquesta pràctica sobre la nota final de pràctiques és del 60%.

1: Execució aplicacions i utilització de senyals en Linux [Del 17d'abril al 2 de maig] (5%+5%+10%+20%+20%)

1.1. Un aspecte important de les aplicacions/processos en Linux és el seu entorn d'execució. Aquest està definit per un conjunto de variables d'entorn (per exemple $PATH, $HOME) que defineixen entre altres coses on buscar els executables dels programes/ordres, la ubicació de l'usuari o l'intèrpret d'ordres que s'està utilitzant.

Les variables d'entorn definides, les podem obtenir mitjançant la comanda *env*. També es pot crear noves variables d'entorn, creant i exportant la variable dins del script utilitzat (en el nostre cas el bash). Per exemple: export TMP_DIR=/tmp

La següent comanda és un exemple de la utilització de les variables d'entorn i altres aspectes de l'intèrpret d'ordres de Linux (bash):

- tar -zcvf $BACKUP_DIR/backup_`date "+%d-%m-%Y"`.tgz $HOME/tmp

Indiqueu, quin seria el resultat de l'execució d'aquesta ordre? Justifiqueu la resposta. ¿Quins passos previs necessaris heu hagut de realitzar, per poder executar aquesta ordre i obtenir els resultats esperats?

La comanda realitza un empaquetat i compressió (mitjançant l'ordre tar -zcvf) de tot el directori *tmp* de la carpeta home de l'usuari ($HOME/tmp). El resultat del empaquetat es guarda en el fitxer *backup_DD-MM-YYYY.tgz* (DD-MM-YYYY és la data actual) en el directori definit mitjançant la variable d'entorn $BACKUP_DIR.

Abans de poder executar la comanda anterior, és necessari crear la variable d'entorn $BACKUP_DIR perquè apunti a un directori (per exemple, $HOME/Backup):

- export BACKUP_DIR=$HOME/Backup

1.2. Indicar les diferències existents entre els modes d'execució de les següents comandes:

- tar -zcvf ./backup.tgz $HOME
- tar -zcvf./backup.tgz $HOME &
- at 24:00 tar -zcvf ./backup.tgz $HOME

La primera comanda executa la comanda *tar* en primer plànol (foreground), de manera que l'usuari pugui interactuar amb l'aplicació. Fins que no es completa l'ordre, l'usuari no podrà llançar una altra execució amb està consola / intèrpret.

La segona comanda s'executa en segon pla (background o en mode batch) i l'usuari no pot interactuar amb ella. Una vegada arrencada l'execució concurrent d'aquesta ordre, l'usuari torna a tenir el control de forma immediata i pot llançar una altra comanda.

La tercera comanda s'executa de forma diferida. L'execució de l'aplicació no es realitza immediatament, sinó que s'espera fins arribar a l'hora/data definida per l'usuari. En l'exemple, fins que no siguin les 12 de la nit l'ordre tar no s'executarà. Igual que ocorre amb l'execució en segon pla, se li retorna de forma immediata el control a l'usuari perquè pugui seguir treballant.

1.3. Estudiar el codi del programa *exec_in.c*, (el font se us inclou amb l'enunciat de la pràctica). Quina funcionalitat té aquest programa? Quines de les ordres de l'apartat anterior proporciona una funcionalitat més semblant a la d'aquest programa?

El programa executa una comanda o una aplicació especificada per l'usuari una vegada hagin transcorregut un determinat temps (especificat en segons).

El programa rep un nombre variable d'arguments. El primer argument (sense tenir en compte el nom de l'aplicació) és el nombre de segons que s'ha d'esperar abans d'executar la comanda diferida. El segon argument és el nom/ruta de l'aplicació a executar. I a partir d'aquest argument podem tenir els diferents paràmetres per a l'aplicació.

L'aplicació el primer que fa és dormir-se durant els segons especificats per l'usuari. A continuació prepara l'ordre d'execució de la comanda, concatenant en una cadena el nom de l'aplicació amb tots els paràmetres de la mateixa especificats per l'usuari. Finalment, invoca a la funció de llibreria *system* que executa l'ordre que li passem i finalitza el programa.

La funcionalitat d'aquesta aplicació és semblada a la de la comanda *at* del sistema.

1.4. Modificar el programa exec_in.c, per obtenir el mateix resultat però reemplaçant les funcions de llibreria sleep i system per crides al sistema. Podeu utilitzar *exit*, *fork*, *exec** entre altres crides al sistema. Heu de utilitzar la comanda *sleep* per esperar que transcorri el temps necessari. El nou programa es dirà exec_in2.c i mantindrà la sintaxi del programa original.

El codi font de la solució està en el fitxer exec_in2Ok.c

1.5. Implementeu una comanda, anomenada *ntimes*, que permeti executar una ordre (cmd1) n vegades de forma concurrent (n<=100). En el cas que es produeixi algun error en l'execució de la primera ordre llavors s'executarà la segona ordre (cmd2) especificat per al tractament d'errors. L'usuari podrà especificar el nombre de vegades que s'executarà la primera ordre, els dos comandaments i tants arguments com es vulgui:

Sintaxi: *ntimes* n <cmd1> [args. ..] --error <cmd2> [args ...]

Exemple:

```
./ntimes 2 date --error touch error.txt
./ntimes 2 sleep 30 --error echo "Error first command"
```

L'usuari pot avortar l'execució dels n ordres mitjançant un Ctrl+\ o Ctrl+4 (senyal SIGQUIT), el que provocarà també l'execució de la comanda d'error.

Observacions:

- El comportament d'aquesta aplicació hauria de ser com el del executable ntimesOk (recordeu que es disposa d'un executable de referència per a cada pas de la pràctica).
- Les n execucions del cmd1 s'han de realitzar de forma concurrent. El cmd2 no s'executa fins que tots els cmd1 s'hagin completat.
- Podeu obtenir la posició de cada un dels ordres dins dels arguments utilitzant del següent codi:

```
int a, cmd2;

for (a=2;a<argc;a++)
{
   if (strcasecmp(argv[a],"--error")==0)
   {
       cmd2 = a+1;
       argv[a]=NULL;
   }
}
```

El codi font de la solució està a l'arxiu ntimes_Ok.c

2: Utilització pipes en Linux [Del 3al 13de maig] (40%, els dos apartats tenen el mateix pes)

2.1. Implementar una variant de la comanda *ntimes* (anomenada *ntimes2*) que permeti redireccionar tota la sortida estàndard de cadascuna de les execucions al seu respectiu arxius (la sortida estàndard de la primera execució es redirigirà al fitxer stdout1, la de la segona execució al fitxer stdout2 i la sortida de l'última al fitxer stdoutN). Tota la sortida d'error de totes les execucions s'enviarà cap a un mateis fitxer (fitxer d'error). Els noms dels arxius s'especificaran mitjançant el paràmetre --out (per especificar el prefix dels arxius de sortida estàndard) i el paràmetre --err (per especificar el nom del fitxer d'error).

Sintaxis: *ntimes2* n <cmd1> [args...] --error <cmd2> [args...] –out <file> --err <file>

Exemple:

```
./ntimes2 2 date --error touch error.txt -out execout
 --err execerr

./ntimes2 5 ls -la --error echo "Error first command"
 --out execout -err execerr
```

Observacions:

- El comportament d'aquesta aplicació hauria de ser com el del executable ntimes2Ok (recordeu que es disposa d'un executable de referència per a cada pas de la pràctica).
- Les n execucions del cmd1 s'han de realitzar de forma concurrent. El cmd2 no s'executa fins que tots els cmd1 s'hagin completat.
- Podeu obtenir la posició de cada un dels ordres dins dels arguments utilitzant del següent codi:

```
Char *outfileprefix=NULL, *errfilename=NULL;
int a, cmd2;
```

```
for (a=2;a<argc;a++)
{
   if (strcasecmp(argv[a],"--error")==0)
   {
       cmd2 = a+1;
       argv[a]=NULL;
   }
   elseif (strcasecmp(argv[a],"--out")==0)
   {
       outfileprefix=argv[a+1];
       argv[a]=NULL;
   }
}
```

- Per crear els noms dels fitxers a partir del prefix (outfileprefix) i el nombre de comanda (h+1), podeu utilitzar la funció *sprintf* que permet donar format a una cadena (com *printf*) però generant el resultat en una cadena que se li passa com a primer argument:

```
sprintf(outfilename, "%s%d", outfileprefix,h+1);
```

El codi font de la solució està en el fitxer ntimes2_Ok.c

2.2. Modificar la comanda *ntimes* perquè ara la sortida per pantalla de cada execució s'imprimeixi amb un color diferent. Per a això, la sortida de les ordres es redirigiran cap al programa principal el qual s'encarregarà d'imprimir-per pantalla amb el color corresponent (podeu utilitzar la funció write_string de l'arxiu rutines.h). La sortida del comandament d'error es mostrarà en vermell.

Sintaxis: *ntimes3* n <cmd1> [args. ..] --error <cmd2> [args ...]

Exemple:

```
./ntimes3 3 date --error touch error.txt
./ntimes3 5 ls -la --error echo “Error first command”
```

Observacions:

- El comportament d'aquesta aplicació hauria de ser com el delexecutable *ntime3ok* (recordeu que es disposa d'un executable de referència per a cada pas de la pràctica).
- La sortida de totes les execucions de les ordres i de la sortida d'error s'ha de mostrar de manera entrellaçada. No és vàlid mostrar primer la sortida dela primera cmd1 i quan estigui acabi mostrar la dela segona cmd1, i així successivament. Per poder realitzar això, la lectura dels descriptors de la pipe s'ha de realitzar de forma no bloquejant. De manera que si en la pipe d'una de les execucions dela cmd1no hi ha informació, es pugui passar a llegir la pipe de la

següent execució dela cmd1 o de la sortida d'error. Per poder activa la lectura no bloquejant cal utilitzar la crida al sistema *fcntl*, passant-li el descriptor que llegirem:

```
/* Activate the non-blocking read for fd */
if ((flags=fcntl(fd, F_GETFL, 0))<0)
    Error("Error reading fd flags.");
if (fcntl(fd,F_SETFL, flags|O_NONBLOCK)<0)
    Error(Error setting fd flags.");
```

- Quan es llegeix un descriptor de fitxer en mode no bloquejant i no hi ha informació la crida al sistema read retorna el codi d'error EAGAIN.

El codi font de la solució està en el fitxer ntimes3_Ok.c

Recursos

Recursos Basics

- Document "Introducció a la programació d'Unix" (disponible al campus virtual). Vegiu la informació relativa a com crear nous processos dins d'una aplicació multiprocés (des de la pàgina 8 fins la pàgina 27, fent èmfasi en les crides al sistema fork, exec, wait i exit). També haureu d'estudiar els mecanismes per a la comunicació de processos mitjançant canonades (Pàgines 54-67).
- Qualsevol manual bàsic de llenguatge C.
- Manual de crides al sistema instal·lat a qualsevol màquina UNIX (comanda man).

Recursos Complementaris:

- Mòduls 6, 7 de l'assignatura: no cal llegir-los de cap a peus, però poden ajudar a entendre què hi ha darrere de les crides al sistema utilitzades.

Criteris de valoració

El pes dels apartats sobre la nota final de la pràctica s'indica al principi de cadascun. El pes de cada apartat es distribueix de forma equitativa entre tots els subapartats que ho integren.

En la correcció es tindran en compte els següents aspectes:

- Les respostes hauran d'estar articulades a partir dels conceptes estudiats en teoria i en la guies de l'assignatura.
- Es valorarà essencialment la correcta justificació de les respostes.
- S'agrairà la claredat i la capacitat de síntesis en les respostes.

- La capacitat d'anàlisi dels resultats obtinguts i la metodologia seguida per a la seva obtenció.
- El codi lliurat ha de poder executar-se correctament en qualsevol màquina amb Linux. Que el codi s'executi correctament en el vostre ordinador, no implica necessàriament que sigui correcte. Revisar diverses vegades la vostra solució i no ignorar els *warnings* que us pugui estar generant el compilador.

Format i data de lliurament

Es crearà un fitxer zip que contingui un fitxer pdf amb la resposta a totes les preguntes i els fitxers .c amb el codi font dels exercicis.

El nom del fitxer tindrà el format següent: "Cognom1Cognom2PRA2.zip". Els cognoms s'escriuran sense accents. Per exemple, l'estudiant Marta Vallès i Marfany utilitzarà el nom de fitxer següent: VallesMarfanyPRA2.tar

La data límit per al lliurament de la pràctica és el **dilluns13de maigde 2019**.

Nota: Propietat intel·lectual

Sovint és inevitable, en produir una obra multimèdia, fer ús de recursos creats per terceres persones. És per tant comprensible fer-ho en el marc d'una pràctica dels estudis del Grau Multimèdia, sempre i això es documenti clarament i no suposi plagi en la pràctica.

Per tant, en presentar una pràctica que faci ús de recursos aliens, s'ha de presentar juntament amb ella un document en què es detallin tots ells, especificant el nom de cada recurs, el seu autor, el lloc on es va obtenir i el seu estatus legal: si l'obra està protegida pel copyright o s'acull a alguna altra llicència d'ús (CreativeCommons, llicència GNU, GPL ...). L'estudiant haurà d'assegurar-se que la llicència que sigui no impedeix específicament seu ús en el marc de la pràctica. En cas de no trobar la informació corresponent haurà d'assumir que l'obra està protegida pel copyright.

Hauran, a més, adjuntar els fitxers originals quan les obres utilitzades siguin digitals, i el seu codi font si correspon.

Un altre punt a considerar és que qualsevol pràctica que faci ús de recursos protegits pel copyright no podrà en cap cas publicar-se en Mosaic, la revista del Graduat en Multimèdia a la UOC, a no ser que els propietaris dels drets intel·lectuals donin la seva autorització explícita.