Vampiria 0.1.1alpha

Guastella Marco alias Vasta

31/10/2015

Sommario

Indice

Capitolo 1

Il framework Vampiria

Vampiria e‘ un framework nato dall'esigenza di velocizzare l'esecuzione di un insieme di processi linux. A runtime Vampiria prende in input un file xml ed in base ad esso genera e gestisce(input,output,signal,variabili d'ambiente,terminazione etc..) un sistema di processi (ogni processo esegue un'applicazione di sistema o un modulo interno al framework).
Vampira ha i seguenti obbiettivi:

1. Creare delle componenti per evitare la riscrittura di codice ridontante di applicazioni dello stesso tipo;
2. Creare dei plugin(insieme di moduli) per testare nuove tecnologie.Per esempio se voglio creare un'applicazione di rete p2p e testarne il funzionamento utilizzo vampiria in questo modo:
   - Creo un plugin ed il relativo modulo;
   - Per la creazione dei plugin utilizzo alcune componenti ed evito di scrivere codice ridondante;
   - Creo un file xml per mandare in esecuzione piu‘ applicazioni dello stesso tipo;
   - Verifico il funzionamento della rete(gli input di ogni singolo modulo vengono passati tramite il file xml);
3. Creare dei sistemi di processi tramite un semplice file xml.

1.1 Esecuzione di Vampiria

A runtime Vampiria prende in input un file xml ed esegue i passi in *Figura 1.1.*

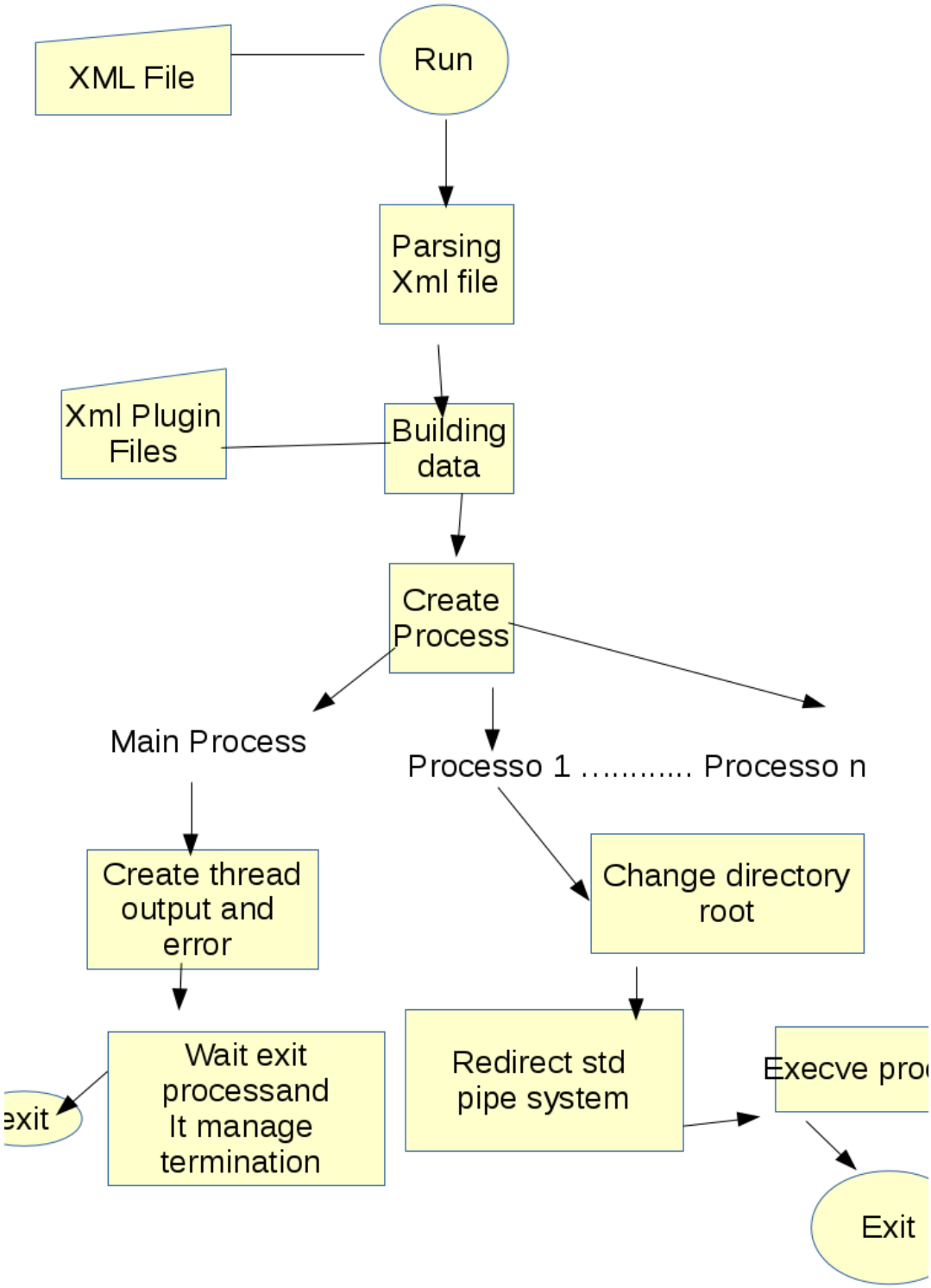

Figura 1.1: sistema

- Parsing xml files: Verifica se la sintassi del file xml e‘ valida e crea un albero;
- Building data: Dall'albero generato dal parsing fa un'analisi semantica dei dati e crea le strutture dati utili per l'esecuzione dei processi.Da notare che in questa fase in input vengono presi dei files xml che servono a verificare la sintassi del modulo da chiamare;
- Create Module: Crea un fork per ogni processo;
- Processi figli: Per ogni processo figlio viene cambiata la directory di lavoro,vengono reindirizzati gli standard [input,output,error] verso le pipe di comunicazione dei processi e viene eseguita la funzione execve() (Per i moduli il comando eseguito VPATH/bin/vmodule);
- Main Process:Il processo principale dopo la creazione dei processi implementa un sistema di threads per la gestione degli stdandard [output,error] dei processi. Questo meccanismo serve ad indirizzare gli output e gli error verso lo schermo,file o pipe di input di altri processi. Da notare che ogni output puo‘ essere mandato in piu‘ destinazioni;
- Wait Terminazione processi: Attende la terminazione dei processi e per ogni processo terminato prima fa alcune operazioni descritte sempre nel file input(ad esempio puo‘ mandare un signal ad un altro processo),libera le risorse (pipe,chiude i threads output e error) e ritorna in fase di attesa fino a quando tutti i processi non sono terminati.(ctrl ĉ termina vampiria)

1.2 Filesystem vampiria

Questa sezione spiega a cosa servono le directory e i file presenti nella root di vampiria:

- la directory bin/ contiene gli eseguibili di vampiria;
  - bin/vampiria e‘ l'eseguibile principale;
  - bin/vmodule serve al framework per eseguire un singolo modulo in una fork(viene chiamato tramite la system call execve().
- la directory cache/ e‘ utilizzata in fase di runtime da vampiria per la creazione delle directory di lavoro dei processi. Ogni sistema eseguito ha una propria directory che contiene le sotto-directory di lavoro dei suoi processi(viene eleminata alla fine dell'esecuzione);

- il file clear.sh serve a resettare il framework;
- cmp/ e‘ la directory delle componenti;
- il file Copyright e‘ la licenza;
- doc/ e‘ la directory della documentazione;
- il file doxygen.conf serve per generare la documentazione;
- dtd/ e‘ la directory dei dtd utilizzati dal parsing xml per verificare la sintassi dei file;
  - il file vampiria.dtd e‘ il dtd del file xml di input del framework;
  - il file module.dtd e‘ il dtd dei file xml utilizzati per la descrizione dei moduli.
- il file install.sh serve per compilare ed installare vampiria;
- lib/ e‘ la directory dei file oggetto delle componenti(cmp.so);
- plugin/ e‘ la directory dei plugin;
- scratch/ e‘ la directory per l'esecuzione;
- src/ e' la directory dei sorgenti;
- src/module e' la directory dei file utilizzati per la creazione dei moduli;(vmp.h deve essere importato da ogni modulo);

1.3 Installazione Vampiria

Per installare Vampiria scaricare il file vmp-0.1.1.tar.gz dal cito http://www.ragnu.it ed eseguire i seguenti passi:

- tar -xvzf vmp-0.1.1.tar.gz;
- cd vmp-0.1.1;
- ./install.sh;(Installare prima tutte le librerie richieste in install.sh)

Per generare la documentazione dei sorgenti di Vampiria eseguire(posizionarsi nella root Vampiria):

- doxygen doxygen.conf;(documentazione in $VPATH/doc/html);

Per disinstallare Vampiria(posizionarsi nella root Vampiria):

- ./clear.sh;

1.4 Run Vampiria

In $VPATH/bin e‘ presente l’eseguibile vampiria usarlo nel seguente modo:

- vampiria -h : stampa l’help;
- vampiria -v : stampa la versione,l’autore e la licenza;
- vampiria -l : stampa la lista dei plugin installati;
- vampiria -r : stampa la root path;
- vampiria -p [plugname] : stampa la lista dei moduli presenti nel plugin;
- vampiria -p [plugname] [modname]: stampa il topic del modulo includendo la sintassi xml da utilizzare per eseguirlo;
- vampiria file.xml: run vampiria;
- vampiria -d file.xml: run vampiria in debug mode.

Capitolo 2

Componenti e plugin

2.1 Creare componenti

Le componenti devone essere scritte in c++.
Un esempio dei file per creare una componente si trova nella directory $VPATH/doc/ex/vcmp_ex.0.
Per creare una componente eseguire i seguenti passi:

- creare la directory $VPATH/cmp/vcmp_{nome_componente}.{Version};
- copiare il file install.sh,cmp.h,Copyright,README dentro la componente;
- creare la directory src/ dentro la componente;
- modificare il file install.sh seguendo i passi descritti dal file;
- modificare i file README e Copyright;
- scrivere i file .cc e .h dentro la dirctory src/;
- scrivere i prototipi delle funzioni o delle classi che possono essere utilizzati dai plugin o da altre componenti in cmp.h;

2.2 Installare componenti

Per installare una componente eseguire i seguenti passi:

- copiare la directory della componente e il suo contenuto (vcmp_ex.0/) in cmp/;

- Entrare nella directory ed eseguire ./install (Installare tutto quello che l'install richiede).

2.3 Creare plugin

I Plugin devone essere scritti in c++.
Un esempio dei file per creare un plugin si trova nella directory $VPATH/-doc/ex/vmp_ex.0.
Per creare un plugin eseguire i seguenti passi:

- creare la directory $VPATH/plugin/vmp_{nome_plugin}.{Version};
- copiare il file install.sh,cmp.h,Copyright,README dentro il plugin;
- creare le directory src/ e xml/ dentro il plugin;
- modificare il file install.sh seguendo i passi descritti dal file;
- modificare i file README e Copyright;
- scrivere i file .cc e .h dentro la dirctory src/;
- Per ogni modulo scrivere un file .cc(esempio $VPATH/doc/ex/vmp_ex.0/src/mod.cc) dentro la directory src/;
- Per ogni modulo creare un file xml modname.xml(esempio $VPATH/-doc/ex/vmp_ex.0/xml/mod.xml) nella directory xml/;

Il file xml per la descrizione del modulo deve avere la seguente forma

<?xml version="1.0" encoding="UTF8"?>
<!DOCTYPE module SYSTEM "../../../dtd/module.dtd">
<module>
<description> descrizione del modulo per il topic</description>
<parameter name="nomeparametro" arg="o|op|zo|zp">descrizione del parametro</parameter>
(lista parametri...)
</module>

Un modulo puo' avere da zero a piu' parametri ed il name per ogni parametro deve essere univoco. L'attributo arg indica quante volte puo' essere ripetuto quel parametro nel file xml principale e ha i seguenti valori:

- "o" : deve essere presente una volta;

- "op" : deve essere presente una o piu‘ volte;
- "zo" : puo’ essere presente zero o una volta;
- "zp" : puo’ essere presente zero o piu‘ volte;

2.4 Installare plugin

Per installare un plugin eseguire i seguenti passi:

- copiare la directory del plugin e il suo contenuto (vcmp_ex.0/) in plugin/;
- Entrare nella directory ed eseguire ./install (Installare tutto quello che l’install richiede).

Capitolo 3

Xml files

Un file input di vampiria ha il formato xml e deve seguire le regole sintattiche del dtd vampiria.dtd. Il file deve essere posizionato in una sottodirectory di scratch/.
Formato file:

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE vampiria SYSTEM "../../dtd/vampiria.dtd">

<vampiria>
<process id="proc1">
..-istruzioni processo1...
</process>
.
.
.
<process id="procn">
..-istruzioni processon...
</process>
</vampiria>

Il tag <vampiria> </vampiria> e' il tag principale dove sono contenuti i processi da eseguire in parallelo (<process id="idname"></process>). I processi non possono avere id uguali.All'interno del tag <process> ci sono le istruzioni per il processo.
Nel proseguio del capitolo sono descritti i tag per le istruzioni del processo in ordine di scrittura.

3.1 import

Il tag <import file="pathfile" name="iname" /> serve per copiare il file "pathfile" nella directory root del processo,prima dell'inizio dell'esecuzione.

- attributo file: pathname del file da copiare(richiesto);
- attributo name: il nome del file nella root directory(richiesto).

Ci possono essere zero o piu' occorrenze di <import>.

3.2 module e system

Il tag <module plugin="plugname" name="modname" debug="true|false"></module> serve ad eseguire un modulo all'interno del processo.

- attributo plugin: il nome del plugin in cui e' situato il modulo(richiesto);
- attributo name: il nome del modulo(richiesto);
- attributo debug: true se il modulo e' in modalita' debug,falso altrimenti(facoltativo default false);
- tag interni:<putenv></putenv> (zero o piu' occorrenze),<parameter></parameter> zero o piu' occorrenze.

Il tag <system cmd="pathcmd"></system> esegue un comando esterno al framework.

- attributo cmd: il comando da eseguire(richiesto);
- tag interni:<putenv></putenv> (zero o piu' occorrenze),<arg></arg> (zero o piu' occorrenze).

Il processo devere avere una sola occorrenza di module o system.

3.2.1 putenv

Il tag <putenv>env</putenv> puo' avere una o piu' occorrenze in <module> o <system> e serve a modificare le variabili d'ambiente per ogni singolo processo.
Esempio:<putenv>LD_PRELOAD=libproxychains.so.3</putenv>

3.2.2 parameter

Il tag <parameter name="nome parametro">value</parameter> puo‘ avere una o piu‘ occorrenze in <module> e serve per gli input del modulo.Il formato dei parametri per ogni modulo e‘ descritto nel file xml di descrizione del modulo;

- attributo name: nome del parametro di input(richiesto);
- value: il valore del parametro;

3.2.3 arg

Il tag <arg>value</arg> puo‘ avere una o piu‘ occorrenze in <system> e serve per argomenti di input del comando esterno.

- value:il valore dell'argomento;

Esempio Comando "/bin/ls -l":

<system cmd="/bin/ls">
<arg>-l</arg>
</system>

3.3 stderr

Il tag <stderr file="filename"/> serve per stampare lo standard error del processo.
Puo‘ avere zero o una occorrenza.Se non e‘ presente lo standard error non e‘ stampato.

- file:la path del file dove stampare lo standard error(facoltativo;se non e‘ presente sara' stampato a schermo);

Esempi:
<stderr /> stampa a schermo lo standard error.
<stderr file="/path/filename" /> stampa lo standard error nel file.

3.4 screen,fileout,pipeout

Il tag <screen/> serve per stampare lo standard output a schermo. Puo' avere zero o una occorrenza.
Il tag <fileout file="nomefile" /> serve a stampare lo standard output in un file.Puo' avere zero o piu' occorrenze:

- attributo file:la path del file dove stampare lo standard output(richiesto);

Il tag <pipeout id="processid" /> serve ad indirizzare lo standard output del processo nello standard input di un'altro processo.Puo' avere zero o piu' occorrenze.

- attributo id:id processo(richiesto);

Il tag <screen> deve essere messo prima degli altri due.I tag <fileout> e <pipeout> possono avere piu' occorrenze a seguire.Lo standard output puo' essere stampato e indirizzato su piu' occorrenze.

3.5 export

Il tag <export name="iname" file="pathfile"/> serve a copiare un file interno alla root directory del processo in un altro file alla fine dell'esecuzione del processo.Puo' avere zero o piu' occorrenze.

- attributo name:il nome del file nella root directory(richiesto);
- attributo file: il nome del file di destinazione(richiesto).

3.6 kill

Il tag <kill id="processid" signal="4" /> serve a mandare un segnale ad un altro processo alla terminazione del processo in cui risiede.Puo' avere zero o piu' occorrenze.

- attribute id:id del processo a cui manda il segnale(richiesto);
- attribute signal: il numero del segnale da mandare(facoltativo default SIGINT).

Capitolo 4

Sviluppi futuri

Questa versione di vampiria pone le basi per lo sviluppo di un framework con i seguenti obbiettivi:

- creare nuove funzionalita‘ per la gestione dei processi o per altre operazioni del core di linux tramite il semplice uso di tag xml;
- integrare componenti e plugin in modo da velocizzare lo sviluppo di applicazioni software;
- creare e distribuire una serie di file xml in modo da rendere semplice l'implementazione di sistemi software complessi;
- superare alcune limitazioni del framework e risolvere bugs;
- creare script d'avvio per le varie distribuzioni linux ed automatizzare l'avvio delle applicazioni;
- tanta fantasia e divertimento.

Un saluto ai gentili lettori e spero che Vampiria possa aiutarvi a rendere piu‘ veloce il vostro lavoro.

Copyright

Author: Marco Guastella alias Vasta
Web page:<www.ragnu.it>
Email: <vasta@ragnu.it>