Relazione fase 0 progetto sistemi operativi

Prerequisiti

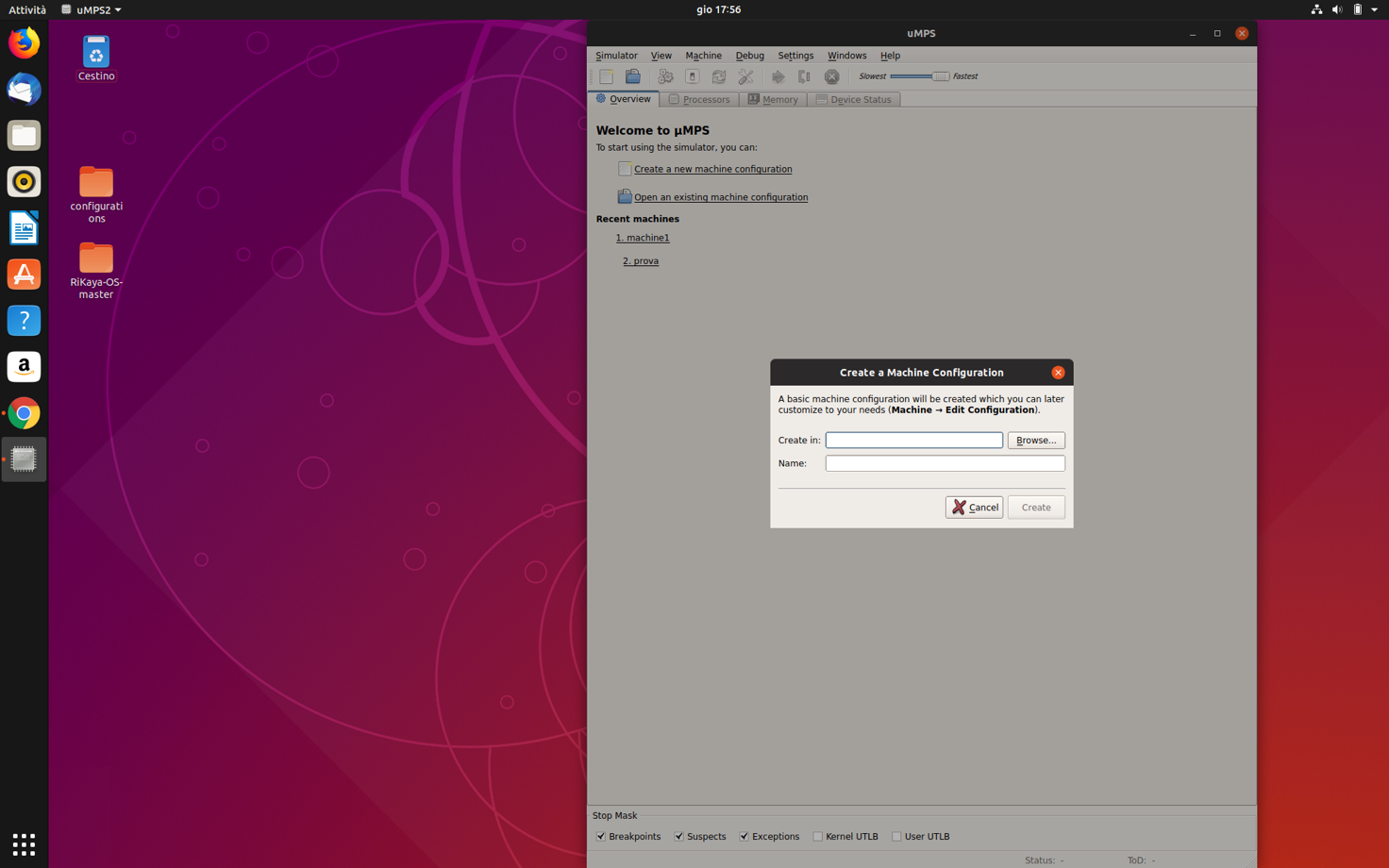
* Scaricare ed estrarre (in una destinazione a piacere) il file Rikaya\_phase0.rar
* Aver scaricato ed installato l’emulatore umps2 nel proprio sistema

Configurazione macchina

Per far funzionare correttamente il traduttore vi è richiesta una serie di configurazioni che verranno illustrate qui sotto. Come prima cosa si avrà la necessità di aprire l’emulatore umps2.

Una volta aperto si potrà procedere con la configurazione vera e propria. Per semplicità andremo a virgolettare le parole ogni qual volta si parli di un menù a tendina.

1. “Simulator” -> New Configuration che servirà a creare una nuova macchina virtuale.



Si aprirà questo form, e nella text box create in andrà inserita la locazione dove verrà salvata la nostra macchina, mentre name sarà il nome di questa.

1. Spostarsi nella cartella Phase0 attraverso il terminale, poi digitare il comando make -f Makefile
2. “Machine” -> Edit Configuration
3. Alla voce **Boot** cliccare *Browse* e cercare il Core file *kernel.core.umps* creato nella cartella *rikaya*  in seguito alla compilazione
4. Alla voce **Debugging Support** cliccare *Browse* e cercare la System Table *kernel.stab.umps* creata nella cartella *rikaya* in seguito alla compilazione.
5. Cliccare nella voce devices, andare nella sezione printers, spuntare la checkbox per la prima stampante e dargli un nome con estensione .umps

Una volta fatto ciò si potrà far partire la macchina virtuale con le sue limitate funzioni.

Come far partire la macchina

Per far partire la macchina bisognerà seguire questi semplici passi:

1. “Machine” ->Power on
2. “Windows” ->terminal0
3. “Debug” ->continue

E da questo momento in poi basterà seguire le indicazioni stampate a terminale per continuare l’esecuzione.

Infine per trovare l’output bisognerà andare dentro la cartella dove abbiamo posizionato la configurazione ed andando ad aprire il file della stampante troveremo il nostro input stampato come le schede perforate.

Relazione codice

Innanzi tutto il nostro programma è costituito da quattro file che sono : main, printer, terminal e punchCard ognuno di essi poi costituito nel file .c e .h .

Per quanto riguarda il main abbiamo una serie di inizializzazioni di variabili e funzioni che serviranno a stampare e leggere da terminale e funzioni che permetteranno di stampare attraverso la nostra stampante.

All’interno del metodo main troveremo un while che ci permetterà di stampare tutte le volte che l’utente ne avrà la necessità anche più volte consecutivamente. All’interno di questo while poi ci sarà una funzione che andrà a leggere l’input da tastiera (chiamata getprint) che come prima cosa andrà a stampare su terminale un aiuto per l’utente , che gli servirà per sapere cosa stampare. Una volta stampato si utilizzerà una funzione chiamata readline che non farà nient’altro che andare ad inserire un buffer (che era stato già creato in precedenza con le varie inizializzazioni) la nostra stringa , fino a quando essa non sarà terminata. Una volta inserita la stringa dentro il buffer si andranno ad inizializzare una serie di variabili come la lunghezza (len) o la progressione della percentuale di stampa (tmpstr) che serviranno per la stampa vera e propria sulla stampante. Una volta inizializzato il tutto si potrà procedere alla traduzione della stringa in schede perforate e poi inviata questa traduzione alla stampante. Per effettuare questa traduzione si andrà ad utilizzare il metodo cstrtopunch che ha come compito smistare le varie parole all’interno del buffer, ovvero se c’è uno spazio si tronca la parola in caso contrario si può procedere alla traduzione vera e propria, e questa traduce i bit di ogni singolo char con “\*” in caso il bit sia 1 mentre traduce in “-“ nel caso il bit sia a 0 poi ritorna il puntatore della lettera tradotta in modo che poi in seguito possa essere stampato. Una volta fatto ciò ciò che viene ritornato dalla traduzione viene inviato alla stampante per venire tradotto, e questo ciclo si ripete finche il buffer non è vuoto, oltre che la stampa vera e propria vengono stampate su terminale delle informazioni sulla progressione della stampa come la percentuale, oppure se quest’ultima termina senza errori viene stampato sempre su terminale il completamento della stampa. Infine per poter riprendere a stampare dopo aver fatto la prima stampa , sempre a terminale verrà richiesto se si vuole proseguire a stampare, e grazie alla lettura su terminale dopo la risposta data dall’utente il programma ripartirà con la lettura della nuova parola o la terminazione.

Invece per il terminale le funzioni utilizzate sono la lettura di una stringa, ove verrà letto carattere per carattere (term\_getchar) e la scrittura di una stringa che poi utilizzerà una funzione ausiliaria che gli permetterà di stampare carattere per carattere (term\_puts e term\_putchar)

Infine per la classe della stampante come si fa anche per il terminale si avrà una funzione che stamperà stringhe ove quest’ultima utilizzerà una funzione ausiliaria che gli permetterà di stampare carattere per carattere. Questa funzione ausiliaria (prin\_putchar) andrà a verificare che lo stato della stampante non sia occupato e nel caso questa sia libera si procederà alla stampa del carattere, ed una volta terminato si manderà il comando di fine stampa (CMD\_ACK) proprio per informare che la stampa è stata ricevuta e quindi andata a buon fine, questa procedura verrà effettuata fino a quando la nostra stringa da stampare non sarà terminata grazie al for che risiede all’interno della funzione che stampa la stringa (prin\_puts)