CODES' NOTE

February 25, 2020

FUCKING CODER

1 SHELL

1.1 Basic Function

The Function of SHELL:
Interpret the user's input and excute these commands.

Shell do three things in its **LifeTime:**

- Initialize
- Interpret
- Terminate

The functions of shell can be divided into three parts:

- Read Command
- Understand Command
- Excute Command

Read Command:
Generally, Shell read in a string and store it into an array.
step1. To read in Command, we need **getchar()** function.
getchar() function read in one single char each time. So we need a loop to make it.
step2. We need to decide when to finish reading.
Typically when we accept an **Enter** or **EOF**, we need to stop reading in.
EOF: end of file. value is -1.

Understand Command:

It means that we need to extract information from strings. Typically, we need to know what program is required, what arguments it has. To simplify our program, we understand the input as

Programm+argumenmts.

It means that the first char should be the name of the program we want to call. The left chars are these required arguments.

These commands and arguments should be divided by space, tab(\t),return(r),newline(n),vertical tab(v).

Namely we have

step1. split string according to there characters. we use **strtok** function to split string. And we also need a loop to iteratively split the string.

step2. store there char into array.

Excuted Commands:

2 My First Own Project: Assistant

The advantage of this CODE:

- If your code is correct, you don't have to check your code every time;
- It would work for different work;
- It would work for different operators, you don't have to wirte code explicitely for each operator. Namely, if you want you can easily generate the code that calculate 12 different kinds of operators.

It's function would include

- code generation (c);
- File management, including encryption, sorting, searching ... and writting assistant;
- Reminder;
- Help derivation;
- Automatic web searching, sorting, saving, analyzing...

This project would be a highly personalize and it will be an assisant of myself.

2.1 Assistant for Contraction

$$C(t) \quad = \quad \Phi^{Bab}_{\alpha\beta}(t')\tau^{bc}_{\beta\gamma}(t',t)\Phi^{Acd}_{\gamma\kappa}(t)\tau^{da}_{\kappa\alpha}(t,t')$$

Conver this line (latex source code) to

Pseudo code:

for a,b,c,d

for $\alpha, \beta, \gamma, \kappa$

for(t')

$C(t) \quad = \quad \Phi^{Bab}_{\alpha\beta}(t') * \tau^{bc}_{\beta\gamma}(t',t) * \Phi^{Acd}_{\gamma\kappa}(t) * \tau^{da}_{\kappa\alpha}(t,t')$

$\Rightarrow$

corr[0][t_src][t]+=vector_products[0][t_snk][c1][c2]*gamma[s1][s2]*peram[t_src][t][s2][s3][c2][c3]*conj(vec

Convert function:

- **Step1.**split line according to *, and = ;
- **Step2**.split token according to bracket ']' and '[';
- **Step3.**split these string in [...] according to $'$,$'$and interpret it as index ;
- **Step4.**Determine repeated indexes;
- **Step5.**Write a **for** loop for each repeated index.

Qusetions:

- **Q1.**Firstly, How to determine which index is repeated;
- **Q2.** How to combine these variables with corresponding indexes.
- **Q3.** How to deal with these indexes that repreated but doesn't need loop over ?
- **Q4.** How to deal with these operators

Solutions:

- **S1**. We should write a simple function to search repeated indexes;
- **S2.** We should sotore variables and correspondings as var$[i][j]$. var$[i][0]$ stores the variable name, var$[i][1-n]$ store the indexes of i-th variable;
- **S3.** Φ has only one time index while τ has two time indexes. But we don't have to repeated with these time indexes, so we don't have to deal with it. Φ looks like $\Phi[p,t,\ldots]$, τ looks like $\tau[t,t',\ldots]$ So we just only need to left first two indexes unchanged.

- **S4.** For simplicity, we don't deal with bracket operaters seperately. Our input only include operators like $+, -, =, *\cdot$ These operaters should be intepreted in a sequence. Firstly, the $'='$. Secondly, $'+'$ and $'-'$. Finally, the $'*'$. In this way, we can know what operator is intepreting. To not forget any operator, we should store operators with correspoinding variables. For example, $a = b + c$. We should interpret it as a, $= b$, $+c$. Genearlly, for a expression like $a = b * f + c - g$
  - **Step1.** Interpret it as a, $= b * f$, $+c$, $-g$;
  - **Step2.** Interpret $= b * f$ as $= b$, $*f$;
  - **Step3.** combine $a, = b, *f, +c, -g$ to be c code; $a = b * f + c - g$. We only need to just combine these operators.
  - **Step4.** Usually, an variable would have several indexes so we need to interpret these indexes. For example, $a[t, t', \alpha, \beta, a, b]$ should be interpret as $a[t][t'][\alpha][\beta][a][b]$. It should be split by [and $'$,$'$.